University of Castilla-La Mancha

Department of Information Technologies and Systems

MARBLE: Modernization Approach for Recovering Business Processes from Legacy Information Systems

---

June 2012

PhD Student:

Ricardo Pérez del Castillo

Advisors:

Ignacio García Rodríguez de Guzmán

Mario Piattini Velthuis

To Elisabeth for sharing side effects of this thesis and her unconditional support.

Acknowledgments

First of all, I would like to express my gratitude to my supervisors Ignacio García and Mario Piattini for their technical and non-technical advice during the realization the Thesis, which would not exist without their supervision.

I wish to extend my thanks to the Alarcos Research Group and all its kind members, for providing me with all the human and technical resources that were necessary to finish this Thesis. My greatest thanks go to all my PhD colleagues.

I am also very grateful to the research groups that have given me the opportunity to carry out different research stays. Firstly, my thanks go to the University of Innsbruck and particularly to Barbara Weber for her great efforts when supervising me. Secondly, my thanks go to the University of Bari and especially to my supervisors there, Danilo Caivano and Teresa Baldassarre.

Last but not least, my most important thanks go to my family for always devoting their attention and support and, in particular, the greatest thanks go to Eli for sharing all my experiences during this time.

Index

ABSTRACT **1**

1. CHAPTER 1. INTRODUCTION **5**

1.1. BACKGROUND AND MOTIVATION 5
1.2. RESEARCH HYPOTHESES AND OBJECTIVES 9
1.3. RESEARCH METHOD 10
1.3.1. Action Research *11*
1.3.2. Research based on Case Studies *11*
1.3.3. Meta-Analysis *13*
1.3.4. The Bender Method *14*
1.4. SUMMARY OF THE THESIS 14
1.4.1. MARBLE *15*
1.4.2. Static Technique *18*
1.4.3. Dynamic Technique *19*
1.5. CONTEXT OF THE THESIS 20
1.6. DOCUMENT STRUCTURE 20

2. CHAPTER 2. STATE-OF-THE-ART **29**

2.1. ARCHITECTURE-DRIVEN MODERNIZATION 29
2.2. KNOWLEDGE DISCOVERY METAMODEL 61
2.3. SURVEY OF MODERNIZATION TOOLS 77

3. CHAPTER 3. RESEARCH PROPOSAL **87**

3.1. OVERVIEW 87
3.2. STATIC APPROACH 111
3.3. DYNAMIC APPROACH 145

4. CHAPTER 4. SUPPORTING TOOL **201**

5. CHAPTER 5. EMPIRICAL VALIDATION **209**

5.1. PRECISION AND RECALL INDICATORS 209
5.2. META-ANALYSIS 221

6. CHAPTER 6. CONCLUSIONS **241**

6.1. SUMMARY 241
6.2. RESEARCH HYPOTHESIS AND ANALYSIS OF GOALS 243
6.3. MAIN CONTRIBUTIONS AND IMPLICATIONS 245
6.4. FUTURE RESEARCH LINES 247
6.5. RESULTS 248
6.5.1. Published Papers *248*
6.5.2. Submitted Papers *253*
6.5.3. Relationship between Publications and Research Themes *254*
6.5.4. References to our Research Work *254*

REFERENCES **259**

Index of Figures

Figure 1. Scenarios to discover and reconstitute business processes embedded in legacy information systems 6

Figure 2. Horseshoe Modernization Model 8

Figure 3. Action research stages 12

Figure 4. An overview of the MARBLE framework 15

Figure 5. Static and dynamic techniques proposed in line with the MARBLE framework 18

Figure 6. Research project timeline during PhD Thesis development 21

Figure 7. Relationship between the chapters and research issues of this Thesis. 24

Figure 8. Timeline of the most relevant contributions 25

Figure 9. Main contribution of Section 3.1 as regards the proposal of the PhD Thesis as a whole 87

Figure 10. Main contribution of Section 3.2 as regards the proposal of the PhD Thesis as a whole 111

Figure 11. Main contribution of Section 3.3 as regards the proposal of the PhD Thesis as a whole 145

Index of Tables

Table 1. Description of research projects in which the PhD Thesis was carried out.....21

Table 2. Summary of research works published during the PhD Thesis.....248

Table 3. Relationship between publications and research issues of the PhD Thesis.....255

ABSTRACT

Context: Business processes have become a key asset in companies, since they allow them to carry out their daily operations, and an appropriate business process management helps companies to improve their competitiveness. Companies' business processes are supported by many enterprise information systems which undergo software erosion over time, thus leading them to become legacy information systems. The volatile IT industry often tempts companies to replace legacy information systems with new ones. However, these systems cannot always be completely discarded because they gradually store a significant amount of valuable business knowledge as a result of progressive maintenance over time. Evolutionary maintenance is the most suitable mechanism to deal with the software ageing problem since it preserves business knowledge. But recovering the underlying business knowledge from legacy systems is first necessary in order to preserve this vital heritage, which is not present elsewhere.

Objective: This thesis proposes and validates MARBLE, a framework and a set of framed techniques with which to recover and rebuild the business process models that are actually supported by legacy information systems. MARBLE, which can be considered as a business process archeology method, makes it possible to preserve the business knowledge buried in legacy information systems, thus allowing it to be effectively used during software modernization.

Method: MARBLE is a generic framework based on ADM (Architecture-Driven Modernization), which uses the KDM (Knowledge Discovery Metamodel) to represent the information retrieved in a common and standard manner. MARBLE proposes a set of models at four different abstraction levels according to the model-driven development principles, along with three model transformations between those levels to progressively reduce the gap between legacy information systems and business process models. This thesis proposes two particular techniques framed in MARBLE. The first technique statically analyses the source code and represents the embedded knowledge in a KDM repository, which is then analyzed by applying a pattern matching technique to discover business process models, while the second technique analyzes the source code at runtime to obtain event logs which are then analyzed to retrieve business processes. The research methodology includes the action research method combined with research based on case studies. The proposed framework has been validated with several case studies involving real-life industrial legacy information systems in different domains (health-care, education, e-administration, and so forth).

Results: MARBLE, together with its supporting tools, proved to be suitable to obtain business process models from legacy information systems with adequate levels of exactness and completeness. Similar results were obtained for all the different MARBLE-framed techniques based on static or dynamic analysis by considering different legacy software artifacts (e.g., source code, event logs, etc.).

Conclusion: The proposed method semi-automatically rebuilds the hidden business processes embedded in legacy information systems. The business process archeology method therefore allows business experts to attain a rapid and meaningful understanding of the organization's business processes. This proposal is less time-consuming and more exhaustive (since it considers the embedded business knowledge) than a manual process redesigned by experts from scratch. As a result, MARBLE provides maintainers with a mechanism with which to modernize legacy information systems in line with the actual business processes of an organization.

CHAPTER 1

Introduction

> *As a consequence of evolutionary maintenance over time, new business knowledge and rules are embedded in legacy information systems.*

1. INTRODUCTION

This PhD Thesis is presented as a collection of published papers. This chapter is therefore devoted to presenting the motivation, description of the research hypotheses and research objectives, along with an explanation of the research methods employed. Finally, a brief summary of this Thesis is presented in order to comprehend the part that each paper presented supports within the Thesis.

1.1. Background and Motivation

In the early 1990s, Business Process Management (BPM) arose as a mechanism with which to maintain and improve the quality of processes and operations carried out by enterprises and organizations (Hammer and Champy, 1993). A business process depicts a sequence of coordinated activities, together with their roles and the data involved, that must be carried out to achieve a business goal (Weske, 2007). Business processes have become a key asset in organizations, since they allow them to carry out their daily operations. Business process management also helps organizations to address technological and organizational changes in order to consequently improve their competitiveness (Castellanos, Medeiros et al., 2009).

Most business processes in organizations are supported by their enterprise information systems. The optimal business process management is therefore achieved when organizations additionally combine the management of their Legacy (existing) Information Systems (LIS) (Jeston, Nelis et al., 2008). The configuration management of legacy information systems is particularly important since these systems undergo a considerable amount of changes during their lifecycles. Legacy information systems change as a result of evolutionary maintenance, in which new business requirements and functionalities are incorporated into the system (Lehman, 1984; ISO/IEC, 2006).

As a consequence of the evolutionary maintenance over time, new business knowledge and rules are embedded in legacy information systems. This embedded business knowledge may not be existent anywhere else (Paradauskas and Laurikaitis, 2006). The evolution of information systems in isolation consequently affects the evolution of business processes (see scenario 1 in Figure 1). In this case, it is necessary to discover and reconstitute the underlying business processes that are currently supported by legacy information systems (Heuvel, 2006).

However, there are many organizations that currently carry out a vast amount of daily transactions through their enterprise information systems without having ever done their own business process modeling. When these organizations deal with business process modeling for the first time, a recurrent method by which to attempt this modeling is the extraction of business processes from legacy information systems (van der Aalst, Reijers et al., 2007) (see scenario 2 in Figure 1). This is owing to the fact that legacy information systems are one of the few

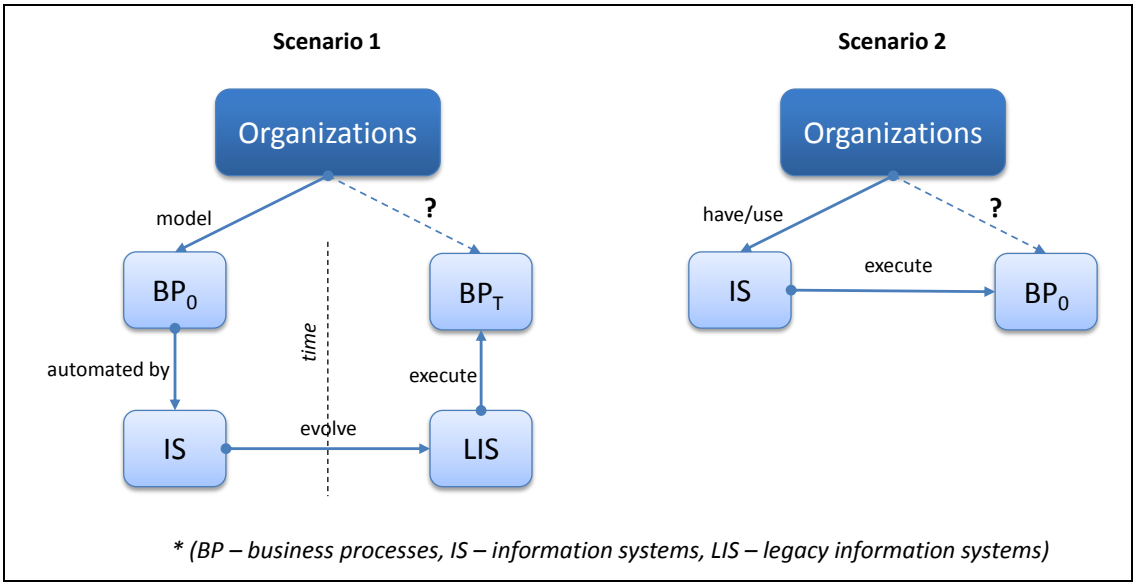


Figure 1. Scenarios to discover and reconstitute business processes embedded in legacy information systems

knowledge assets in organizations that can be used to attain an accurate understanding of the actual business process.

In both scenarios (see Figure 1), retrieving an up-to-date version of business processes from legacy information systems allows organizations to take advantage of at least two main benefits:

- **Benefits for business process modeling.** Business processes can always be up-to-date. Organizations may therefore conduct business process management by following the continuous improvement process (Davenport, 1993). This kind of business process management facilitates an agile adaptation of business processes to meet all the changes that occur in the uncertain environment of a company. The rapid evolution of business processes allows organizations to maintain, or even improve, their degree of competitiveness (Weske, 2007; Jeston, Nelis et al., 2008).
- **Benefits for legacy information systems.** Legacy information systems may continue to be modernized on more occasions. A recent study by the SEI (Software Engineering Institute) states that it is first necessary to retrieve embedded business knowledge in order to modernize systems in line with the organization's business processes (Lewis, Smith et al., 2010). Organizations can thus modernize their legacy information systems whilst they align the new systems with their actual business processes. Legacy information systems are therefore evolved rather than being immediately retired and the ROI (return of investment) on such systems is improved. This is because the lifecycles of these systems are extended, which saves costs as regards new developments from scratch (The Standish Group, 2010).

There are two trends or approaches in literature that can be used to recover embedded business processes. In the first approach, several business process mining techniques and algorithms have been proposed (Medeiros, van Dongen et al., 2004; van der Aalst and Weijters, 2005; Medeiros, Weijters et al., 2007; Rozinat, Jong et al., 2009; Motahari-Nezhad, Saint-Paul et al., 2011). Process mining takes event logs recorded during the execution of enterprise information systems as input. All these approaches assume the presence of event logs and they are not concerned with obtaining event logs from systems. In this respect, other works attempt to obtain event logs from different kinds of Process-Aware Information Systems (PAIS). For example, (Günther and van der Aalst, 2007) provide a generic import framework with which to obtain event logs, while (Ingvaldsen and Gulla, 2008) focus on Enterprise Resource Planning (ERP) systems to obtain event logs from the SAP's transaction data logs.

Unfortunately, the main problem of business process mining is that there are no well-proven techniques with which to obtain event logs from traditional legacy information systems (i.e., non-process-aware information systems) without an in-built logging mechanism. This is an important challenge since there are a vast amount of legacy information systems in industry that do not explicitly support business processes (i.e., non-process-aware information systems). The provision of techniques with which to obtain event logs from traditional legacy information systems may facilitate the application of all the efforts (techniques and algorithms) made in the business process mining field (Pérez-Castillo, Weber et al., 2010).

In the second approach, several proposals consisting of the analysis and inspection of different software artifacts have been proposed in literature. In this Thesis, these kinds of approaches have been termed as Business Process Archeology (Pérez-Castillo, de Guzmán et al., 2011). While business process mining relies on event logs produced at runtime, business process archeology analyzes different artifacts (e.g., source code, databases, or even event models) by means of reverse engineering in order to obtain the underlying embedded business processes.

Some of these kinds of proposals are based on the static analysis of source code. One of these proposals is that of (Zou and Hung, 2006), which statically analyses the source code and applies a set of heuristic rules to discover business processes. (Paradauskas and Laurikaitis, 2006) present a framework to recover business knowledge through the static analysis of data stored in databases. (Ghose, Koliadis et al., 2007) propose a set of text-based queries in source code and documentation with which to extract business rules. The major disadvantage of these proposals (which solely use static analysis as a reverse engineering technique) is that run-time information is ignored. Other methods using dynamic analysis are therefore proposed. (Di Francescomarino, Marchetto et al., 2009) discover business processes through the execution of graphical user interfaces in Web applications. (Cai, Yang et al., 2009) propose an approach that combines requirement reacquisition (to first provide a set of use cases) with dynamic analysis techniques (tracing the system according to these use cases).

The main weakness of the second approach, business process archaeology, is that most efforts are *ad hoc* proposals, which are developed for particular platforms, technologies and specific contexts. This lack of formalization and standardization leads to another challenge related to the automation of such techniques, and the repeatability of business process archaeology techniques in large-scale projects is therefore in doubt (Canfora, Di Penta et al., 2011). In fact, a 2005 study (Sneed, 2005) states that 50% of reengineering projects (based on reverse engineering techniques) fail owing to the lack of standardization and automation, which often leads to overruns in costs.

Standardization and automation challenges limit the applicability of the aforementioned techniques to large and complex legacy information systems. These challenges can be addressed by Model-Driven Development (MDD) principles, i.e., (i) considering and treating all software artifacts as models which conform to specific metamodels, and (ii) establishing automatic transformations between models at different abstraction levels.

The Architecture Driven-Modernization (ADM) initiative (also known as software modernization) launched by the OMG, particularly advocates carrying out a reengineering process by following model-driven development principles. Traditional reengineering consists of three main stages (Chikofsky and Cross, 1990): (i) reverse engineering, which analyzes elements of existing systems and their relationships and obtains an abstract representation of the system; (ii) the restructuring stage, which changes some internal features in the abstract representation of the system (maintainability, understandability, performance, etc.) whilst the external behavior is preserved; and finally (iii) the forward engineering stage, which obtains the target, modernized system by decreasing the degree of abstraction. The ADM initiative incorporates model-driven development principles into traditional reengineering in an attempt to solve standardization and automation challenges. This is known as the horseshoe modernization model (see Figure 2).

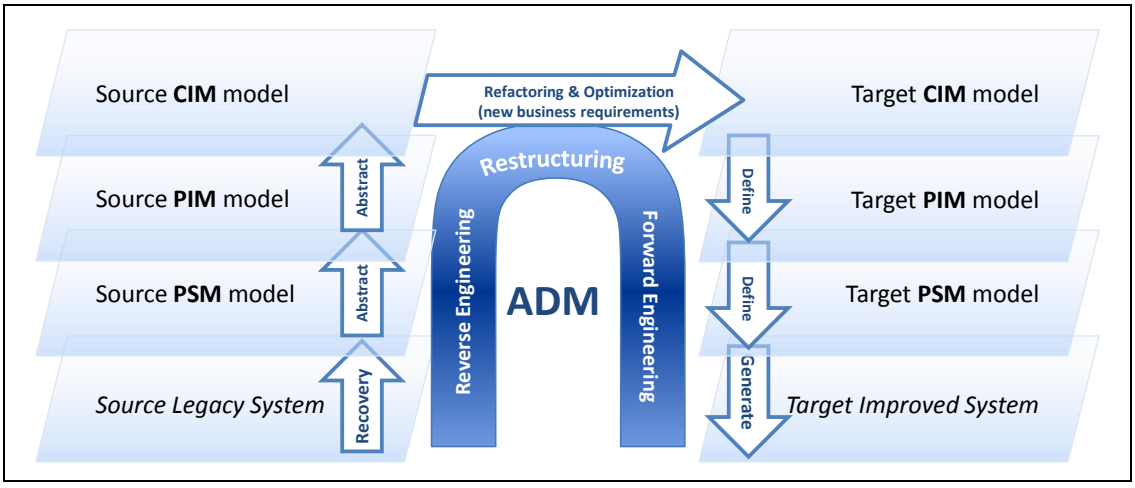


Figure 2. Horseshoe Modernization Model

ADM solves the formalization problem since it represents all the artifacts involved in the reengineering process as models, which are represented in accordance with specific metamodels. The Meta Object Facility (MOF), a standard adopted from the OMG, is used for this purpose to provide both a metadata management framework and a set of metadata services to enable the development and interoperability of model and metadata driven systems. MOF has made a significant contribution to some of the core principles of both the MDA and ADM.

ADM therefore treats all software artifacts homogenously, i.e., as models that can be transformed into other models by using deterministic transformations. These transformations can be formalized , for example, by means of the QVT (Query / Views / Transformations) standard proposed by the OMG (OMG, 2008). The model transformations can consequently be automated through their formalization. Furthermore, the model-driven development principles make it possible to reuse models used in different modernization projects, since a computational independent model (CIM) model can be transformed into several platform independent models (PIM), and each PIM model can in turn be transformed into several platform specific models (PSM) (see Figure 2).

1.2. Research Hypotheses and Objectives

Since the discovery and reconstitution of business processes from legacy information systems can be considered as a reverse engineering method, the main hypothesis of this research is:

HYPOTHESIS

It is feasible to construct a framework by using the ADM approach to retrieve business processes embedded in different kinds of legacy information systems in a standardized and automatic manner.

With regard to the research hypothesis, we believe that ADM can address the standardization challenge since it treats all the artifacts involved in the recovery process homogeneously, i.e., as models. The automation challenge can therefore be addressed thanks to the automated transformations together with the reuse of models at different abstraction levels. The main objective of this Thesis is, therefore:

OBJECTIVE

To define an ADM-based framework with which to discover and reconstitute business processes from legacy information systems.

The attainment of the main objective will be based on the achievement of the following partial objectives:

O1. To carry out an in-depth study of the business process mining and archeology field in the context of software modernization which will provide the state-of-the-art of this Thesis.

O2. To define business process archeology techniques with which to extract embedded business knowledge form different software artifacts (both static and dynamic techniques)

O3. To establish a mechanism with which to generate event logs from traditional legacy information systems in order to reuse existing business process mining techniques.

O4. To represent and integrate all the business knowledge analyzed in a common and platform-independent model repository to be used to retrieve business process models.

O5. To establish a model transformation between the common knowledge repository and the target business processes and define traceability methods between elements transformed at different abstraction levels in order to facilitate feature location in business process models.

O6. To propose a technique with which to discover cross-cutting business process views according to different aspects or features of the legacy information systems.

O7. To develop software tools that will support and semi-automate the techniques proposed.

O8. To conduct an empirical validation of all the techniques through the use of the supporting tools by applying the techniques to real-life information systems in industrial case studies.

O9. To carry out a meta-analysis of the combined results obtained from all the case studies in order to establish measures and indicators with which to assess the proposed techniques in comparison to similar existing ones.

1.3. Research Method

Various research methods with which to achieve the goals established in the main research hypothesis have been employed in this Thesis. Action Research is the main methodology used

throughout this Thesis. Action Research is a qualitative research method based on iterative efforts (action) in order to progressively increment a knowledge base (research) (cf. Section 1.3.1). Apart from this qualitative research method, other empirical research methods such as research based on case studies are used to validate all the techniques proposed (cf. Section 1.3.2). Quantitative research methods are also used to obtain strengthened conclusions by aggregating the results of different case studies. Meta-analysis is a technique which statistically analyzes the combination of results from different case studies (cf. Section 1.3.3). Finally, the Bender method, based on the probabilistic and statistic theory, is used to obtain indicators of measures from the results obtained in the cases studies (cf. Section 1.3.4).

1.3.1. Action Research

Action research is assumed by organizations or researchers with the aim of improving their strategies, practices, and knowledge of the environments within which they practice (Avison, Lau et al., 1999). Action research is a qualitative research method that is widely used in the software engineering field (Santos and Travassos, 2009). Action research considers two main entities: the customer of the research and the scientific community. These entities are often opposed to each other, and the main challenge of the action research method is therefore to satisfy the demands of both entities.

Action research is an iterative process which is oriented towards the progressive addition of new research knowledge that entails an added-value or benefit for the research customer (Wood-Harper, 1985). The iterative process of action-research consists of four stages (see Figure 3). The first stage is *planning*, in which researchers define research question to be answered. In this Thesis, the planning stage consists of the definition of hypotheses and objectives. The second stage is *action*, in which researchers conduct the fieldwork regarding the research questions in real environments. The third stage is *observation*, in which researchers collect the data obtained after the action stage. Finally, the fourth stage is *reflection*, in which researchers analyze the results observed after their actions. The *reflection* stage leads to the generation of new knowledge with which the research customers are provided, and it is used to redefine the research question in the next iteration (see Figure 3).

The research knowledge base is progressively populated with new knowledge in each iteration. The addition of new research knowledge provides research customers with new theoretical knowledge (e.g., models, process, techniques, etc.). However, this knowledge is often proposed without an empirical validation, which generates a gap between theoretical research and its practical application (Moody, 2000). This PhD Thesis therefore also follows a research method based on the conduction of case studies.

1.3.2. Research based on Case Studies

A case study is an empirical enquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not

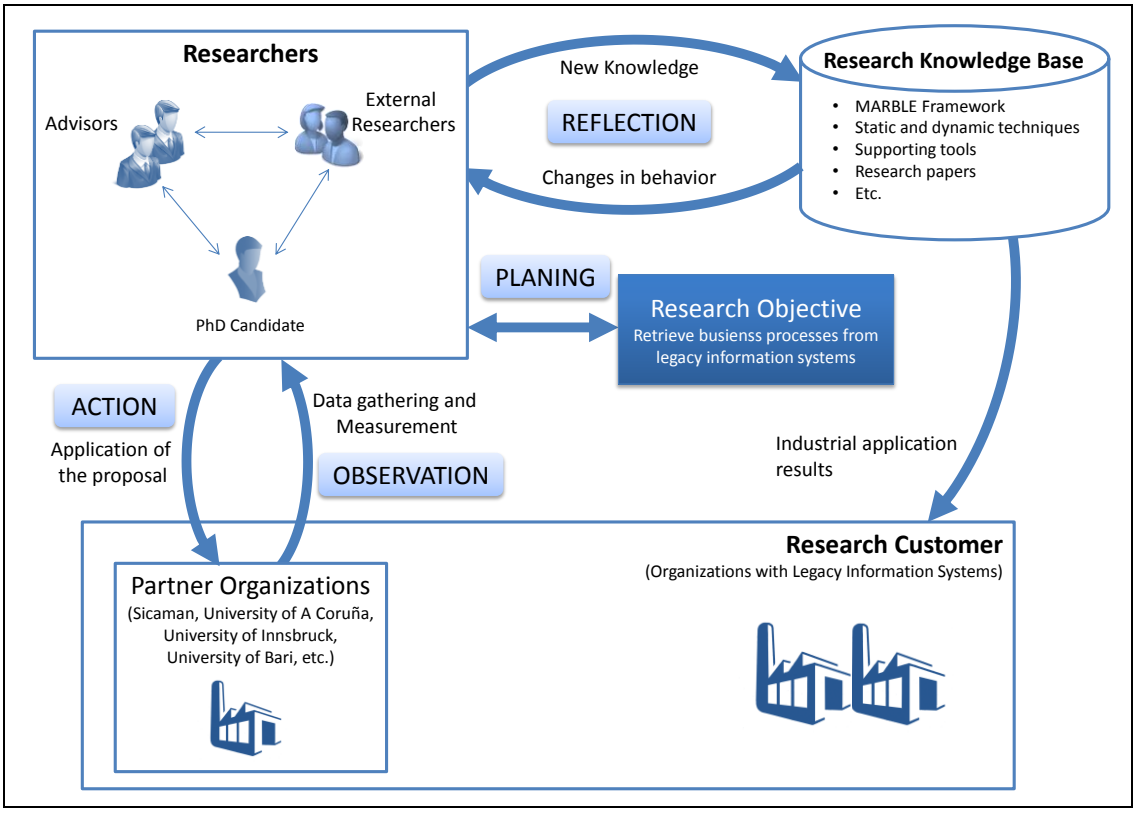


Figure 3. Action research stages

clearly evident (Yin, 2003). While experiments follow a solely quantitative approach, case studies are conducted by collecting qualitative information which is rich but lacks standardized structure and analysis methods. The nature of a case study is inimitable, and such studies are thus difficult to repeat with other cases. The most feasible subjects to validate with case studies are therefore particular projects, processes, artifacts, etc. (Soares Cruzes, Dybå et al., 2011).

This nature signifies that case studies are sometimes criticized for being impossible to repeat and generalize from, for being biased by researchers, and so on (Runeson and Höst, 2009). For example, cases studies do not generate the same results when they are repeated, unlike analytical and controlled empirical experiments. However, the main advantage of case studies is that they provide a deeper understanding of the phenomena under study.

In order to mitigate this problem, some formal methodologies have been proposed with which to plan and conduct case studies in the software engineering field in a similar way to that which has occurred in other disciplines (e.g., social sciences or medicine). The case studies presented in this Thesis have been planned and conducted according to the well-proven methodology proposed by (Runeson and Höst, 2009). Moreover, this methodology is usually aided by the protocol and checklist for conducting software engineering cases studies provided by (Brereton, Kitchenham et al., 2008). The steps defined in the aforementioned methodology are the following:

1. **Background.** This stage identifies previous research on the subject topic in order to establish the main research questions being addressed by the case study.
2. **Design.** This stage first identifies the object of the study (e.g., a new technique, feature, method, etc.) and the purpose of the study through the evaluation of certain properties (e.g., the effectiveness, efficiency, scalability, etc.). These properties are evaluated through the definition of certain measures which are used as dependent variables. Finally, this stage establishes the independent variable of the study by answering the following questions: *is the study a single-case or a multiple-case? Is it an embedded (with various analysis units) or a holistic study?*
3. **Case selection.** This stage establishes criteria with which to select a suitable case under study.
4. **Case study procedure**. This defines a plan with which to execute the case study by establishing execution steps, roles involved in each step, etc.
5. **Data collection**. This identifies the data to be collected and defines a data collection plan with which to record all the relevant data generated during the study execution.
6. **Analysis and interpretation**. This stage inspects all the collected data and identifies criteria with which to interpret the case study findings. Data analysis obtains the evidence chains needed to answer research questions.
7. **Validity evaluation**. The validity of the results is assessed in order to verify that they are unbiased and valid for the whole population to which researchers wish to generalize the results. In addition to detecting threats to the validity, specific actions should be proposed to mitigate all the threats detected.

1.3.3. Meta-Analysis

Case studies usually lack statistical significance and their results are difficult to generalize to a large population. Several statistical methods (e.g., meta-analysis, significance level combination or vote counting) are therefore used by researchers to accumulate and interpret a set of results obtained from different empirical studies that are inter-related because they verify similar hypotheses. In this Thesis, we have used meta-analysis (Miller, 2000; Pickard, 2004) to strengthen the conclusions obtained from all the case studies conducted.

While simple analysis attempts to evaluate a sole study in isolation, meta-analysis combines the results of several. Meta-analysis is a set of statistical techniques that combine the different effect size measures (or treatment effect) of various individual studies. There are several metrics with which to obtain this value, such as the mean difference or correlation coefficients, among others (Hedges and Olkin, 1985). The objective is to obtain a global effect, the treatment effect, of all the cases studies.

Since effect size measures may originate from different environments and heterogeneous studies or experiments, Meta-analysis first obtains standardized measures of each one. The global effect size is then obtained as a weighted average of standardized measures, in which the

most commonly used weights are the sample size or the standard deviation. Finally, together with the estimation of the global effect size, meta-analysis provides an estimated confidence interval and a *p-value* which is used to decide the meta-analysis hypotheses.

1.3.4. The Bender Method

In addition to meta-analysis, other methods are also necessary to obtain indicators concerning the measures used in case studies. Indicators allow researchers to interpret how good a particular value of a measure is. Indicators are important when there are no benchmark values for a measure in a particular area.

In this Thesis we use the *Bender* method (Bender, Parker et al., 1989) to extract thresholds for defined measures. Thresholds help us to assess the numerical results of these measures, thus making it probable that adequate results will be found. This method was developed to discover thresholds in epidemiological studies. The *Bender* method assumes that the risk of an event occurring is constant below a specific value (i.e., the threshold) and increases according to a logistic equation. The *Bender* method defines a Value of an Acceptable Risk Level (VARL) where p_0 is the probability of an event occurring (Eq. 1). The p_0 value is indicated by researchers and can vary from 0 to 1. For example, p_0 = 0.6 indicates that there is a probability of 0.6 for considering the measure under study as appropriate. On the other hand, α and β are coefficients of the logistic regression equation, which is shaped as (Eq. 2).

$$VARL = \frac{1}{\beta} \cdot \left(ln\left(\frac{p_0}{1 - p_0} - \alpha \right) \right) \quad (1)$$

$$y = \alpha + \beta \cdot measureX \quad (2)$$

The independent variable in the logistic regression equation is the measure or measures (denominated as *measureX* in (Eq. 2)), from which researchers wish to extract their thresholds. The dependent variable must be a binary variable and is usually called a *goodness* variable. This dependent variable indicates whether the independent variables are considered good by taking into account certain factors such as error degree, suitability, and so on.

The *Bender* method usually works by extracting different VARLs with the specific values of$p_{0,2}$, $p_{0,4}$, $p_{0,6}$, $p_{0,8}$. These four values of probability divide the measure distributions into five groups, which are often associated with linguistic labels for an accurate interpretation (very low, low, medium, high, and very high).

1.4. Summary of the Thesis

In order to achieve the main research goal, this PhD Thesis proposes MARBLE (*Modernization Approach for Recovering Business processes from LEgacy systems*). MARBLE is a reverse engineering framework based on ADM which is used to discover and retrieve

business processes embedded in legacy information systems. MARBLE is a generic framework that provides standard guidelines based on ADM specifications to develop methods and techniques with which to retrieve business process models. In fact, together with MARBLE (the generic framework), this Thesis proposes two particular techniques: a business process archaeology technique based on the static analysis of source code, and a business process mining technique that considers system execution information.

1.4.1. MARBLE

Since MARBLE is an ADM-based framework, it uses the KDM (Knowledge Discovery Metamodel) (ISO/IEC, 2009) to represent all the knowledge involved, the cornerstone specification of the ADM initiative (see Figure 4). The KDM specification defines a standard metamodel that facilities an integrated representation and management of all the knowledge extracted by reverse engineering from all the different software artifacts of legacy information systems. MARBLE extracts legacy knowledge which is integrated into a common KDM repository and it then gradually transforms this knowledge into business process models. The abstraction gap between software artifacts (such as source code) and business process models must be reduced progressively. MARBLE therefore defines four kinds of models at four different abstraction levels. MARBLE additionally establishes, according to the ADM initiative, three model transformations between the four abstraction levels (cf. Section 3.1).

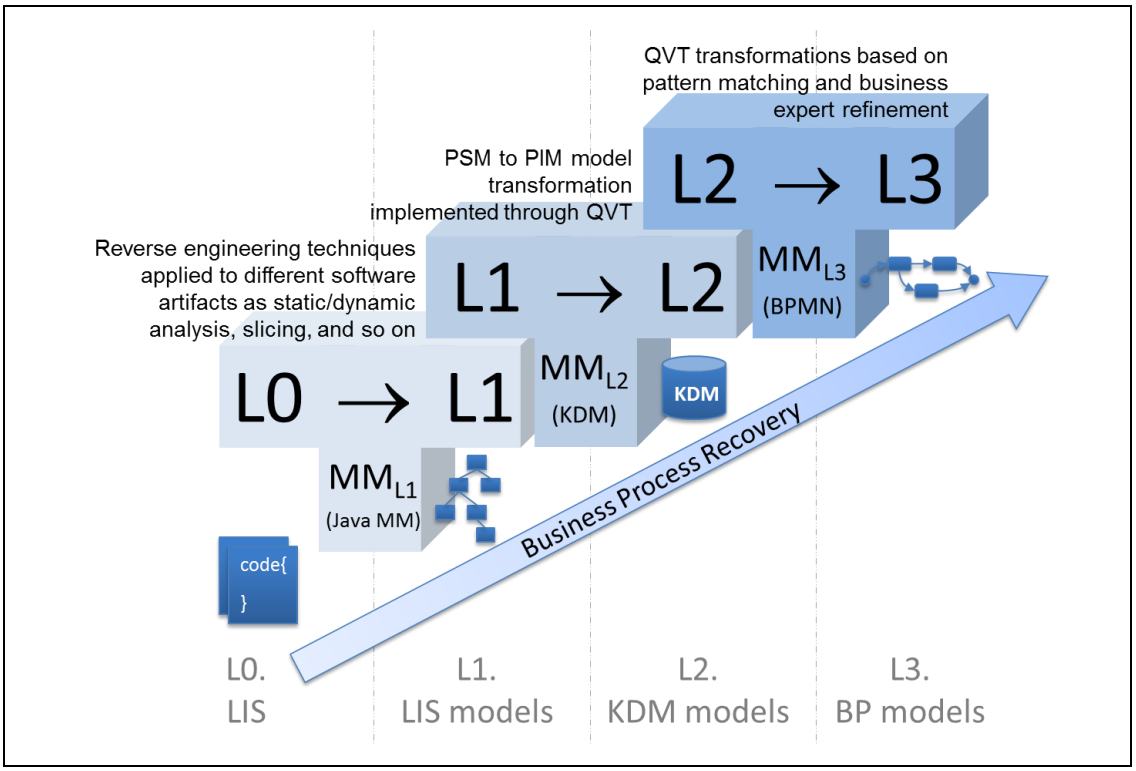


Figure 4. An overview of the MARBLE framework

The four generic abstraction levels proposed in MARBLE are the following:

- **Level L0**. This is the lowest level of abstraction. L0 represents the legacy information system in the real world as a collection of different software artifacts (e.g. source code, database, documentation, etc.).
- **Level L1**. This level consists of several specific models, i.e., one model for each different software artifact involved in the archeology process, such as source code, database, user interface, and so on. These models are considered to be PSM models since they depict the software artifacts according to their specific technology or platforms.
- **Level L2**. This level consists of a common PIM model which represents the integrated view of the set of PSM models at L1. The standard KDM metamodel is used for this purpose, since it makes it possible to model all the artifacts of the legacy system in an integrated and technological-independent manner. Firstly, we say that the L2 model is obtained in an integrated manner because L2 works as a common KDM repository that can be progressively populated with more and more knowledge extracted from an organization's various information systems. This is a key advantage since business processes are usually executed by multiple, heterogeneous systems within an organization. Secondly, we say that L2 is represented in a technological-independent manner because the KDM standard abstracts all those details concerning the technological viewpoint (e.g. the program language).
- **Level L3**. This is the highest level of abstraction which represents a computational independent model of the system. L3 depicts the business processes retrieved from the knowledge concerning legacy information systems represented in the KDM repository at L2. Business process models at L3 are represented according to the BPMN (Business Process Model and Notation) metamodel (OMG, 2009).

MARBLE additionally defines the following three model transformations between the four abstraction levels in order to incrementally obtain business process models (at L3) from legacy information systems (at L0) (cf. Section3.1):

- **Transformation L0-to-L1**. The first transformation obtains PSM models from each legacy software artifact. Traditional reverse engineering techniques (Canfora, Di Penta et al., 2011) such as static analysis, dynamic analysis, subsystem decomposition, and so forth, are used to discover and extract knowledge from a software artifact and build the respective PSM model (see Figure 4). Each PSM model is represented according to a specific metamodel, e.g., a Java metamodel can be used to model source code, or an SQL metamodel can be used to represent the database schema of a legacy information system.
- **Transformation L1-to-L2**. The second transformation consists of a set of model transformations with which to obtain a common PIM model based on the KDM metamodel. This PIM model is built from the PSM models from level L1. The L1-to-L2 transformation can be implemented by means of QVT scripts. These QVT scripts

translate instances of metaclasses of respective metamodels used at L1 and metaclases of the standard KDM metamodel. The transformation from the legacy information system (at L0) to the KDM model (at L2) is made through the level L1 since in many cases, the platform-specific knowledge at the intermediate level L1 might be used to infer more business knowledge at L2. The semantic gap between the legacy information system and the respective KDM model is thus reduced incrementally.

- **Transformation L2-to-L3**. The last transformation is based on a set of business patterns and a matching technique with which to identify them. When a particular structure (according to a pattern) is detected in the KDM model (at L2), that pattern indicates the respective structure of elements that must be built into the business process model (at L3). MARBLE framework implements this pattern matching technique through a model transformation also written in QVT. The L2-to-L3 transformation can additionally be aided by business experts who know the organization. Experts provide external information to refine the first sketch of the business processes obtained after pattern matching. For example, business experts can detect and remove inconsistent or incoherent pieces of business processes; they can refactor the business process models to fit processes to the real behavior of the organization; or they can add manual business activities that are not embedded in legacy information systems.

MARBLE can be used as standard guidelines to provide concrete techniques with which to obtain business processes from legacy information systems. Each concrete technique is characterized by (i) the legacy software artifacts that are considered at L0; (ii) the reverse engineering technique used to extract embedded business knowledge from these artifacts; (iii) the set of business patterns employed in the last transformation; and (iv) whether business expert post-intervention does or does not take place.

This PhD Thesis provides two concrete techniques, which are summarized in Figure 5 and are presented in detail in Sections 3.2 and 3.3 respectively. Although both techniques consider legacy source code as the input software artifact, these techniques address business process recovery from two different perspectives. On the one hand, the first technique statically analyzes the source code and represents the embedded knowledge in a KDM repository, which is then analyzed by applying a pattern matching technique to discover business process models. On the other hand, the second technique analyzes the source code at runtime to obtain event logs that are then analyzed in order to retrieve business processes. While the first technique follows the business process archaeology approach (cf. Section 3.2), the second follows that of business process mining (cf. Section 3.3).

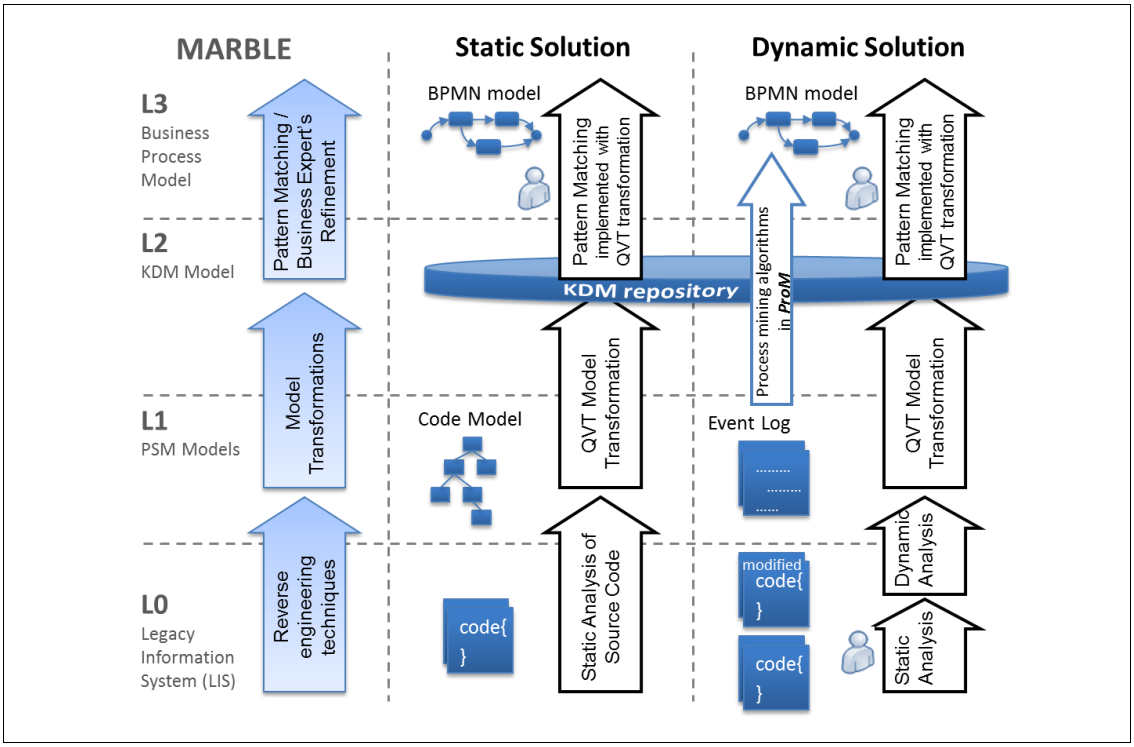


Figure 5. Static and dynamic techniques proposed in line with the MARBLE framework

1.4.2. Static Technique

The static technique framed in MARBLE considers legacy source code as the key software artifact at L0 and static analysis as a reverse engineering technique with which to extract the embedded knowledge at L0 and represent it at L1. The static analysis of source code consists of the sequential, syntactic inspection of the all source code files.

This technique is supported by a tool that has been especially developed for this purpose. The tool has a module with which to analyze source code files (Java files in particular) and builds an abstract syntax tree of the source code, i.e. a code model at level L1 (see Figure 5). The advantage of the static approach is that the syntactic parser that is used to analyze the source code is easy and non-time-consuming to build. Moreover, the tool implements a QVT model transformation to transform the code model into a KDM model at L2. The KDM model provides a standard inventory of all the source code elements and their relationships, and can therefore be used for other business process recovery techniques framed in MARBLE or any other software modernization activity.

The static technique (see Figure 5) provides a set of business patterns to support the discovery of business processes at L3 (Pérez-Castillo, García-Rodríguez de Guzmán et al., 2010). The set of patterns is divided into three categories:

- **Structural patterns**, which deal with the structural elements (e.g., business activities or tasks) and their combinations (such as sequence flows or gateways). There are four patterns which transform: (i) packages, compilation units or other aggregation units (e.g. Java classes or interfaces) into business process diagrams; (ii) methods into tasks; (iii) calls between methods into sequence flows between the respective tasks; and (iv) conditional branching, e.g. *if-then-else* or *switch* statements, into exclusive gateways that branch the main sequence flow.
- **Data patterns**, which deal with data objects and how these objects are related to other structural elements. There are two data patterns, which transform: (i) read program variables for a method into a data object with an association to the respective task; and (ii) written program variables into data objects with an association from the task.
- **Event patterns**, which build all the elements involved in the event management. There are three patterns, which transform: (i) the start method into a start event and sequence flow to the respective task; (ii) end tasks into sequence flows from such tasks to an end event; and (iii) conditional calls into sequence flows with an intermediate conditional event.

The pattern recognition and generation of business process models is implemented in the tool developed by means of QVT transformations (Pérez-Castillo, García-Rodríguez de Guzmán et al., 2010). The tool can also be used by business experts to refine the business processes discovered, since it provides graphical model editors for both KDM and business process models.

1.4.3. Dynamic Technique

The dynamic technique framed in MARBLE (see Figure 5) considers knowledge derived from system execution. Hence, the reverse engineering technique considered in the L0-to-L1 transformation combines static analysis and dynamic analysis.

The static pre-analysis injects statements into certain places of the source code to register execution events in a log when these statements are executed (see Figure 5). Each event recorded in the log specifies the execution of an underlying business task supported by a certain piece of source code. According to (Pérez-Castillo, Weber et al., 2010), the injection of these tracing statements into the source code therefore entails five key challenges that must be addressed: (i) process definitions are implicitly described in legacy source code and, thus, it is not obvious which events should be recorded in the event log; (ii) the granularity of callable units of an information system and activities of a business process often differs; (iii) legacy code not only contains business activities, but also technical aspects which have to be discarded from target business processes; (iv) since traditional systems do not explicitly define processes, it is necessary to establish when a process starts and ends; (v) finally, owing to the missing process-awareness, it is not obvious how the business activities and process instances executed should be correlated.

This technique partially addresses these challenges by considering information provided by business experts and system analysts. This information is necessary to reduce potential noise in the event log caused by the aforementioned challenges. Although manual intervention by experts might appear to be a time-consuming task, this task is less time-consuming and even less error-prone than business process modeling from scratch.

After static pre-analysis, the dynamic analysis records events during system execution (see Figure 5). When the instrumented code is executed, the injected statements invoke a function that records the respective event in the event log. The event log can then be used to discover the business processes by taking the system execution information into account. In the same manner as the static technique, this dynamic technique transforms the event log at L1 into a KDM model at L2, and then transforms this model into business process models at L3. However, this technique reuses existing process mining techniques such as those supported by *ProM*tool (Van der Aalst, Van Dongenm et al., 2009) in order to discover business process models (at L3) from event logs generated after dynamic analysis (see Figure 5).

1.5. Context of the Thesis

This section presents the research projects that constitute the framework in which this PhD thesis was carried out. Figure 6 shows the timeline of the research projects that partially supported this Thesis.

The research addressed in this PhD Thesis has been developed in the context of the following six research projects. Table 1 provides the description of the supporting research projects. The abbreviation of the sponsor organizations that funded these projects are: JCCM (Regional Government of Castilla-La Mancha); MICIN (Spanish Research Ministry); FEDER (European Funds for Regional Development) and CDTI (Center for the Development of Industrial Technology).

Finally, it is worth mentioning that this PhD thesis has also been developed thanks to the research fellowship granted to the author of the Thesis by the FPU (*Formación de Profesorado Universitario*) Program of the *Spanish Education Ministry*.

1.6. Document Structure

The remainder of this PhD Thesis is organized as follows. Chapter 2 summarizes the state-of-the-art and work related to the research topic. Chapter 3 presents a detailed description of the MARLE framework, which is the main proposal of this Thesis, in addition to the static and dynamic techniques framed in MARBLE. Chapter 4 shows the supporting tool developed to semi-automate the proposal. Chapter 5 provides the results of the empirical validation of the research proposal. Finally, Chapter 6 shows a discussion of conclusions, implications and future work of this PhD Thesis.

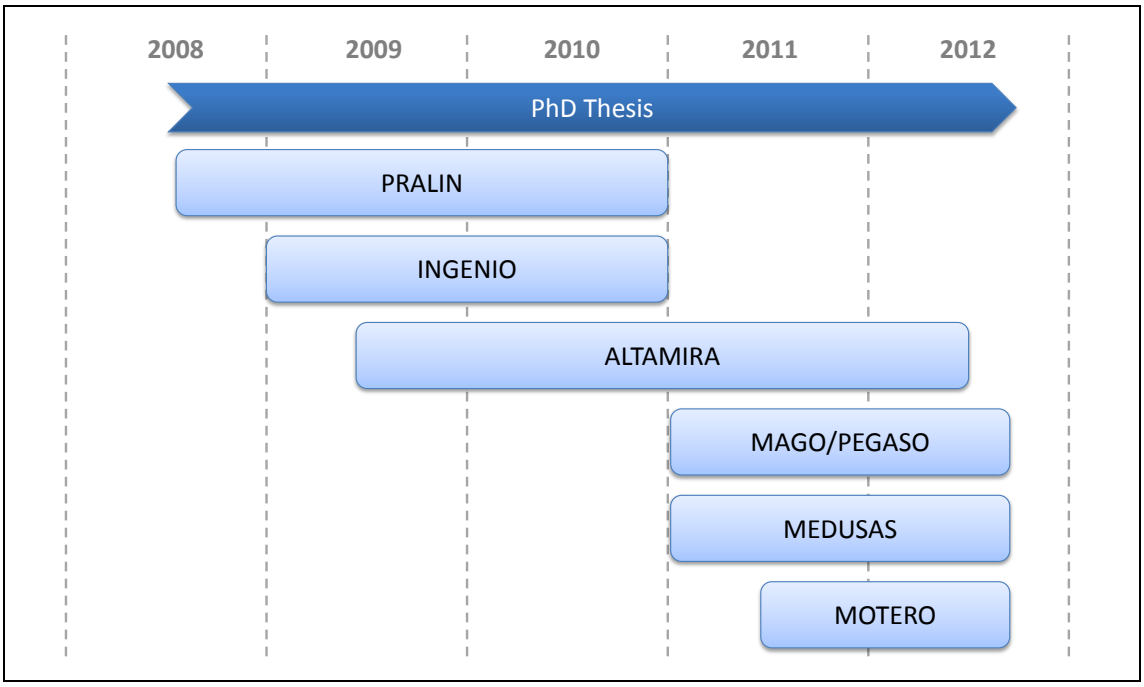


Figure 6. Research project timeline during PhD Thesis development

Table 1. Description of research projects in which the PhD Thesis was carried out

Project	Goal	Sponsored by	Grant (€)	Duration
PRALIN	Testing in software product lines	JCCM	140,000	1/1/2008 31/12/2010
INGENIO	Application of software engineering practices to improve business processes.	JCCM	130,000	1/1/2008 31/12/2010
ALTAMIRA	Application of techniques for modeling and quantitative management of processes to improve the maturity level of software factories	JCCM	240,000	1/4/2009 1/4/2012
MAGO/ PEGASO	Advanced improvement of global software processes.	MICINN and FEDER	489,100	1/10/2009 31/12/2012
MEDUSAS	Improvement and assessment of software design, usability, security and maintenance.	CDTI	140,000	1/1/2009 1/1/2013
MOTERO	Software modernization to obtain software product lines from legacy information systems.	JCCM and FEDER	120,000	01/04/2011 31/3/2013

Since this Thesis is composed as a collection of papers, all the chapters (with the exception of the introduction and conclusion) consist of one or more published research papers to cover the purpose of each chapter.

Eleven of the published research papers have been selected to be part of the core chapters of this PhD thesis owing to both their scientific contribution and the relevance of these papers. Bearing the selected papers in mind, the content of what remains of this document is organized as follows. The name of each chapter is provided, along with a brief description of it and the reference of the paper associated with the chapter.

Chapter 2. State of the Art. This chapter provides the background of the Thesis and introduces all the key terms, such as legacy information system, software modernization, business process, and so on. This chapter is divided into three subsections, which are supported by three different research papers:

- Section 2.1. Architecture Driven Modernization

  Pérez-Castillo, R., I. García Rodríguez de Guzmán, and M. Piattini, Architecture-Driven Modernization, in Modern Software Engineering Concepts and Practices: Advanced Approaches, A.H. Dogru and V. Bier, Editors. 2011, IGI Global. p. 75-103.

  This is a book chapter that presents the ADM initiative, its precursors, its current standards, and the main implications. It also introduces how ADM is used to modernize legacy information systems through business process recovery techniques.

- Section 2.2. Knowledge Discovery Metamodel

  Pérez-Castillo, R., I.G.-R.deGuzmán, and M. Piattini, Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems. Computer Standards & Interfaces, 2011. 33(6): p. 519-532. Impact factor: 0.825.

  This is a journal paper which focuses on the use of the KDM standard to modernize legacy information systems.

- Section 2.3. Survey of Modernization Tools

  Pérez-Castillo, R., I.G.-R.deGuzmán, M. Piattini, and C. Ebert, Reengineering Technologies. IEEE Software Magazine, 2011. 28(6): p. 13-17. Impact factor: 1.511.

  This is a journal paper that presents a survey of common current software modernization tools.

Chapter 3. Proposal. This is the core chapter of the Thesis since it presents the research proposal in detail. This chapter is divided into three sections: the generic MARBLE framework which is supported by one paper; the particular static technique framed in MARBLE which is supported by a second paper; and the dynamic technique which is supported by two additional papers.

- Section 3.1.Overview

  Pérez-Castillo, R., I.G.-R.deGuzmán, and M. Piattini, Business Process Archeology using MARBLE. Information and Software Technology, 2011. 53: p. 1023–1044. Impact factor: 1.507.

This is a journal paper that depicts the MARBLE framework. It provides a detailed explanation of the abstraction levels of MARBLE and the model transformations defined between them.

- Section 3.2. Static Approach

Pérez-Castillo, R., I. García Rodríguez de Guzmán, M. Piattini, and Á.S. Places, A Case Study on Business Process Recovery using an E-Government System. Software Practice & Experience Journal, 2011: 42(2): p. 159–189. Impact factor: 0.573.

This journal paper provides a particular technique framed in MARBLE which is based on the static analysis of legacy source code. This paper also demonstrates its applicability through an industrial case study with an e-government system.

- Section 3.3. Dynamic Approach

Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, and M. Piattini, Process Mining through Dynamic Analysis for Modernizing Legacy Systems. IET Software Journal, 2011. 5(3): p. 304–319. Impact factor: 0.671.

Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, and M. Piattini, Generating Event Logs from Non-Process-Aware Systems Enabling Business Process Mining. Enterprise Information System Journal, 2011. 5(3): p. 301–335. Impact factor: 0.786.

Both journal papers present details of a dynamic technique framed in MARBLE, which obtains event logs from the execution of instrumented versions of legacy information systems. While the first paper introduces a preliminary version of the dynamic technique, the second presents an improved version of the technique. Both papers provide empirical validation of the technique through two industrial case studies applied to an author management system and a healthcare evaluation system, respectively.

Chapter 4. Supporting Tool. This chapter provides the relevant implementation details regarding the supporting tool developed to instrument the static technique previously presented in Section 3.2. This chapter is supported by the following conference paper.

Pérez-Castillo, R., M. Fernández-Ropero, I. García Rodríguez de Guzmán, and M. Piattini, MARBLE. A Business Process Archeology Tool, in 27th IEEE International Conference on Software Maintenance (ICSM'11). 2011, IEEE Computer Society: Williamsburg, Virginia, USA. p. 578-581. ERA CORE A .

Chapter 5. Validation. This chapter focuses on the empirical validation of the research proposal. Although most of the papers presented in the proposal (cf. Chapter 3) contain industrial case studies, this chapter addresses the empirical assessment as a whole by aggregating individual results.

- Section 5.1. Precision and Recall Indicators

Pérez-Castillo, R., L. Sánchez-González, F. García, M. Piattini, and I. García Rodríguez de Guzmán, Obtaining Thresholds for the Effectiveness of Business Process Mining, in 5th ACM/IEEE

International Symposium on Empirical Software Engineering and Measurement (ESEM'11). 2011, IEEE Computer Society: Banff, Alberta, Canada. p. 453-462. ERA CORE A.

This conference paper applies the Bender method in order to obtain indicators, i.e., reference values, with which to measure the precision and recall used in all the case studies conducted. Indicators are necessary to compare the results obtained from different case studies and different techniques.

- Section 5.2. Meta-Analysis

Pérez-Castillo, R., J.A. Cruz-Lemus, I. García-Rodríguez de Guzmán, and M. Piattini, A Family of Case Studies on Business Process Mining using MARBLE.Journal of Systems and Software, 2012. 86(6): p.1370–1385. Impact factor: 1.277.

This journal paper combines the results obtained from all the empirical studies by applying meta-analysis techniques in order to integrate and obtain strengthened conclusions.

Figure 7 summarizes the document structure as regards the parts of the proposal and research topics of this PhD Thesis. Figure 7 will be used at the beginning of each section to provide an overview of its scope.

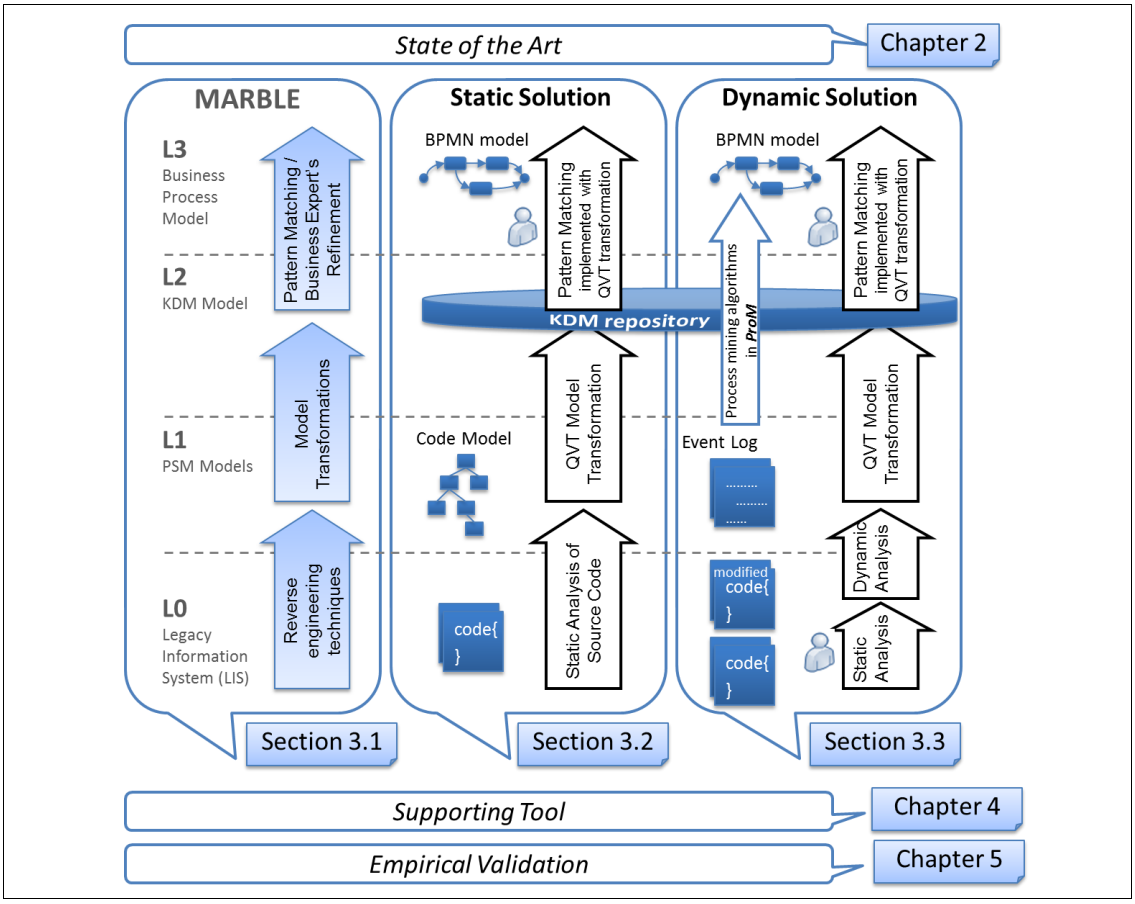


Figure 7. Relationship between the chapters and research issues of this Thesis.

Apart from the publications included in this Thesis, there are more contributions that conform with some of its research objectives. Figure 8 shows a graph depicting the relationships between the contributions included in this document (black nodes) and other preliminary, precursor publications (gray nodes). The different research lines within this Thesis are noted with different kinds of edges (i.e., the static and dynamic technique framed in MARBLE, the empirical validation of these techniques, along with other contributions related to the state-of-the-art) (see Figure 8).A detailed list of all the publications can be found in sub-section 6.5.

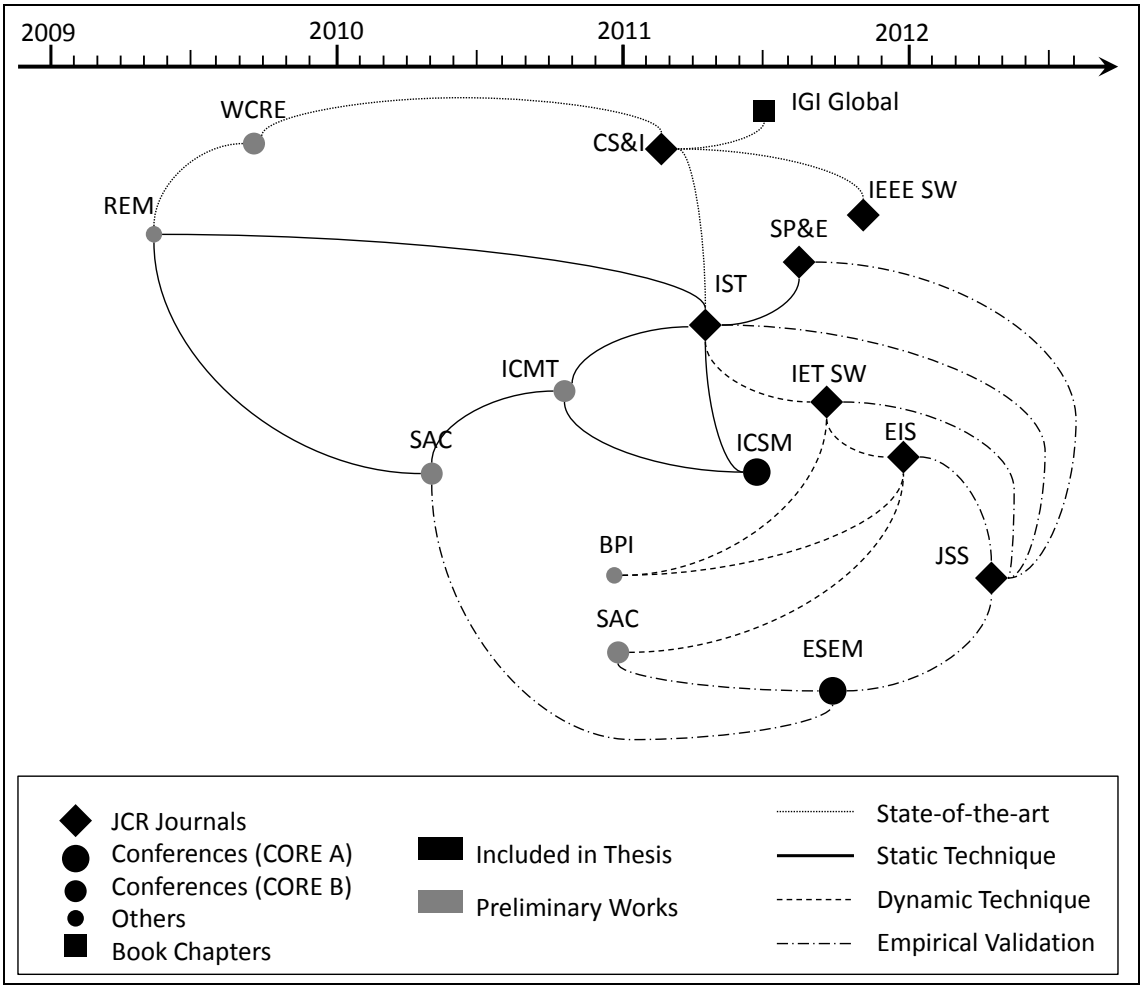


Figure 8. Timeline of the most relevant contributions

CHAPTER 2

State of the Art

> *The KDM metamodel allows to represent all the software artifacts recovered during reverse engineering techniques at different abstraction levels.*

2. STATE-OF-THE-ART

This chapter provides the background to the Thesis and introduces all the key terms, such as legacy information system, software modernization, business process, and so on. This chapter is divided into three subsections.

Firstly, Section 2.1 provides an overview of the standard specifications provided by the Architecture-Driven Modernization Initiative, which can be used to retrieve business processes in a software modernization context. Secondly, Section 2.2 particularly focuses on the KDM specification, which provides a Knowledge Discovery Metamodel to represent the business knowledge extracted from legacy information systems. Finally, Section 2.3 provides a brief survey of software modernization tools related to the research issues.

2.1. Architecture-Driven Modernization

This section provides a detailed description of the ADM initiative, its precursors, its current standards, and the main implications. It also introduces how ADM is used to modernize legacy information systems through business process recovery techniques. This section consists of a book chapter published by *IGI Global* editorial in 2011. The reference to this chapter is: *Pérez-Castillo, R., I. García Rodríguez de Guzmán, and M. Piattini, Architecture-Driven Modernization, in Modern Software Engineering Concepts and Practices: Advanced Approaches, A.H. Dogru and V. Bier, Editors. 2011, IGI Global. p. 75-103.*

Chapter 4
Architecture-Driven Modernization

Ricardo Pérez-Castillo
University of Castilla-La Mancha, Spain

Ignacio García Rodríguez de Guzmán
University of Castilla-La Mancha, Spain

Mario Piattini
University of Castilla-La Mancha, Spain

ABSTRACT

Legacy information systems can be a serious headache for companies because, on the one hand, these systems cannot be thrown away since they store a lot of valuable business knowledge over time, and on the other hand, they cannot be maintained easily at an acceptable cost. For many years, reengineering has been a solution to this problem because it facilitates the reuse of the software artifacts and knowledge embedded in the system. However, reengineering often fails due to the fact that it carries out non-standardized and ad hoc processes. Currently, software modernization, and particularly ADM (Architecture-Driven Modernization), standardized by the OMG, is proving to be an important solution to that problem, since ADM advocates carrying out reengineering processes taking into account the principles and standards of model-driven development. This chapter provides an overview of ADM and shows how it allows legacy information systems to evolve, making them more agile, preserving the embedded business knowledge, and reducing maintenance costs. Also, this chapter presents the software archeology process using ADM and some ADM success stories.

INTRODUCTION

According to the Lehman's first law, an information system must continually evolve or it will become progressively less suitable in real-world environments (Lehman et al., 1998). Indeed, companies count on a vast number of large legacy systems which are not immune to software erosion and software ageing, i.e., legacy information systems that become progressively less maintainable (Polo et al., 2003). Nevertheless, software erosion is due to maintenance itself and the evolution of

DOI: 10.4018/978-1-60960-215-4.ch004

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

the system over time. It is possible to measure this erosion using different metrics (Visaggio, 2001), e.g., dead code, clone programs, missing capacities, inconsistent data and control data (coupling), among others.

The successive changes in an information system degrade its quality, and thus, a new and improved system must replace the previous one. However, the wholesale replacement of these systems from scratch is risky since it has a great impact in technological, human and economic terms (Koskinen et al., 2004; Sneed, 2005). The technological and human point of view is affected since replacement would involve retraining all the users in order to understand the new system and the new technology, or the new system may lack specific functionalities that are missing due to the technological changes. Moreover, the economic point of view is also affected since the replacement of an entire legacy system implies a low Return of Investment (ROI) in that system. In addition, the development or purchase of a new system could exceed a company's budget.

For example, let us imagine a transmission belt in a car engine which deteriorates over use and over time. When this piece is damaged, it must be replaced, and then the engine operates normally. This example is easy, but an information system used in a company is more difficult. When this system ages, it cannot simply be replaced by another new system for two important reasons: (i) a belt costs a few dollars while an enterprise information system costs thousands of dollars, but in addition (ii) the aged system embeds a lot of business knowledge over time that is lost if it is replaced, thus the company with a new system may not operate normally like the car engine.

When companies are faced with the phenomenon of software erosion, evolutionary maintenance is a better solution to obtain improved systems, without discarding the existing systems, thus minimizing the software erosion effects. Evolutionary maintenance makes it possible to manage controllable costs and preserves the valuable business knowledge embedded in the legacy system, since 78% of maintenance changes are corrective or behaviour-preserving (Ghazarian, 2009).

Over the last two decades, reengineering has been the main tool for addressing the evolutionary maintenance of legacy systems (Bianchi et al., 2003). Reengineering preserves the legacy knowledge of the systems and makes it possible to change software easily, reliably and quickly, resulting in a maintenance cost that is also tolerable (Bennett et al., 2000).

The reengineering is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form [...] This may include modifications with respect to new requirements not met by the original system.(Chikofsky et al., 1990)

Nevertheless, a 2005 study states that over 50% of reengineering projects fail (Sneed, 2005). This is due to the fact that in most cases the reengineering usually has two main problems when dealing with specific challenges at this point in time:

- the reengineering of large complex legacy information systems is very difficult to automate (Canfora et al., 2007), therefore the maintenance cost grows significantly.
- the traditional reengineering processes lacks formalization and standardization (Kazman et al., 1998), and thus different reengineering tools that address specific tasks in the reengineering process cannot be integrated or reused in different reengineering projects.

For these reasons, the software industry is demanding reengineering processes that enable the evolutionary maintenance of legacy systems in an automatic and standardized way. The typical reengineering concept has shifted to so-called Architecture-Driven Modernization (ADM) as a solution to those demands.

ADM is the concept of modernizing existing systems with a focus on all aspects of the current system's architecture and the ability to transform current architectures to target architectures. (OMG, 2006a)

ADM advocates carrying out reengineering processes following the MDA (Model-Driven Architecture) standard (Miller et al., 2003), i.e., it becomes possible to model all the legacy software artifacts as models and establishes model transformations between the different MDA abstraction levels. The model-driven principles assist in solving the standardization and automation problems inherent in traditional reengineering processes. In this way, ADM addresses the software erosion phenomenon by modernizing legacy systems. This software modernization can be considered as a kind of software evolution. Its main purposes are (OMG, 2007a):

- Software improvement
- Interoperability
- Migration
- Refactoring
- Restructuring
- Enterprise application integration
- MDA migration

In addition to software erosion management, ADM provides other important benefits such as (OMG, 2007a):

- Revitalizing legacy information systems, making them more agile.
- Reducing maintenance and development costs
- Extending the useful life of legacy information systems, also improving the ROI of those systems.
- Easy integration with other systems and other environments like Service-Oriented Architecture (SOA).

The objective of this chapter is to provide an overview of the emerging concepts and standards related to ADM, and show the way in which these elements are able to solve a typical problem in the software industry: modernizing legacy and ageing systems in companies.

The rest of this paper is organized as follows. The second section summarizes the state-of-the-art of software modernization as well as the most important works related to the concepts in this chapter. The third section gives a detailed account of software evolution by means of the possibilities offered by ADM. Following this, the fourth section presents software archeology as an emerging stage within software modernization processes, focusing on understanding and recovering specific knowledge from existing software assets. The fifth section shows a concrete real world case study in order to clarify the concepts presented previously. The sixth section briefly presents some projects and tools where the ADM approach has successfully been applied. Finally, the last section presents a summary of the chapter and future and emerging trends related to software modernization.

BACKGROUND

This section discusses the main concepts related to this chapter in detail: firstly, it defines legacy systems and their problems; secondly, it presents the reengineering process and its stages; and finally, it shows the ADM approach, its history and the standards that it defines.

Legacy Information Systems

Most companies have existing information systems considered to be *legacy systems* because the code in these systems was written long ago and may now be technologically obsolete. According to (Paradauskas et al., 2006), *"a legacy information system is any information system that significantly resists modification and evolution*

to meet new and constantly changing business requirements". (Ulrich, 2002) states that "*legacy systems are defined as any production-enabled software, regardless of the platform it runs on, language it's written in, or length of time it has been in production"*. (Hunt et al., 2002) go further to state that the *"code becomes legacy code just about as soon as it's written"*.

Software vendors have cultivated a belief that *"anything new is beautiful and that everything old is ugly"* and we have become *"victims of a volatile IT industry"* (Sneed, 2008). Despite the fact that legacy systems may be obsolete, this kind of system usually has a critical mission within the company and represents a valuable asset for companies, since legacy systems embed a lot of business logic and business rules that are not present elsewhere (Sommerville, 2006). The business knowledge embedded in the legacy systems results from the fact that companies maintain their legacy systems over time. This maintenance adds increasingly more functionalities to legacy systems, supporting the company's operation and activities. Therefore, the companies cannot discard their legacy systems, although for this reason, they must deal with the underlying problems of software erosion in the legacy systems (Paradauskas et al., 2006):

- Typically, legacy systems are implemented with obsolete technology which is difficult and expensive to maintain.
- The lack of documentation leads to a lack of understanding of the legacy systems, making the maintenance of these systems a slow and expensive process.
- A great effort must be made to integrate a legacy system together with other systems in the company, since the interfaces and boundaries of the legacy system are not usually well defined.

Reengineering

Two decades ago, the reengineering of legacy systems became one of the most successful practices for dealing with software erosion problems in this type of system. A reengineering process does not discard the whole system, and the enhanced system that it obtains preserves most of the business knowledge embedded in the legacy system. Therefore, the reengineering processes make it possible to carry out evolutionary maintenance of the legacy systems assuming low risks and low costs (Sneed, 2005).

The reengineering process consists of three stages: reverse engineering, restructuring and forward engineering (Chikofsky et al., 1990). The first stage is the reverse engineering process which analyzes the legacy system in order to identify the components of the system and their interrelationships. Then, the reverse engineering stage builds one or more representations of the legacy system at a higher level of abstraction. The second stage is the restructuring process which takes the previous system's representation and transforms it into another at the same abstraction level. This stage preserves the external behavior of the legacy system. Finally, the third stage of the reengineering process is forward engineering, which generates physical implementations of the target system at a low abstraction level from the previous restructured representation of the system.

According to (Kazman et al., 1998), the whole reengineering process can be viewed as a horseshoe model. The reverse engineering stage that increases the abstraction level represents the left side of the horseshoe; the restructuring stage preserves the same abstraction level, thus it represents the curve of the horseshoe; and the last part on the right side of the horseshoe is the forward engineering stage that instantiates the target system.

Applications of Reengineering

Reengineering practices have been applied to several fields in addition to software engineering, such as the hardware industry and business process management. Specifically, the software industry has successfully applied reengineering to many domains. Some examples of these domains are the following:

- Traditionally, reengineering has been used to carry out technological changes or migrations to another program language. One example is the transformation of a legacy system that was developed following a structured paradigm into an object-oriented system. This reengineering process also involves rewriting the code written in a structured programming language such as COBOL or C into other object-oriented language such as JAVA or C++.
- Reengineering has also been used to modify the design of a legacy system in order to improve its maintainability, efficiency, and other system features. For example, imagine a legacy information system based on JAVA technology. In the reverse engineering stage we could analyze the legacy source code and recover the design of the legacy systems, i.e., a set of class diagrams that depict the system. After that, in the restructuring stage, we can modify the design of the systems by adding well-defined design patterns such as the patterns proposed by (Gamma et al., 1995), which improve the maintainability of the system. Finally, the forward engineering stage generates the new source code from the improved class diagrams.
- Another broad field where reengineering has been applied within the software industry is database reengineering. In this respect, (Hainaut et al., 1996) propose reengineering as a mechanism to design relational databases. While the forward engineering of a database requires conceptual design, logical design and physical design, the reengineering approach only needs data structure extraction and data structure conceptualization. This approach makes it possible to migrate towards a relational data model from any kind of data model.

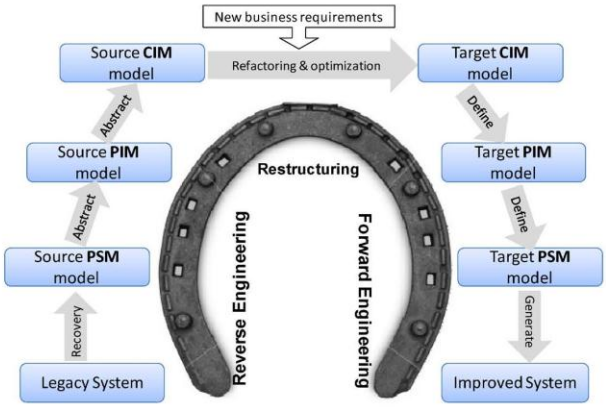
Limitations of Reengineering

Reengineering has been a successful practice in the software industry, but currently more than half of the traditional reengineering projects fail when dealing with specific challenges (Sneed, 2005). The most important problems are the standardization and automation of the reengineering process.

Firstly, standardization constitutes a problem since the reengineering process is typically carried out in an *ad hoc* manner. Thus reengineering projects must focus their efforts on a better definition of the process. Furthermore, the code cannot be the only software asset that the standardization covers, since *"the code does not contain all the information that is needed"* (Müller et al., 2000). The reengineering process must be formalized to ensure an integrated management of all of the knowledge involved in the process such as source code, data, business rules, and so on.

Secondly, automation is also a very important problem. In order to prevent failure in large complex legacy systems, the reengineering process must be more mature and repeatable (Canfora et al., 2007). In addition, the reengineering process needs to be aided by automated tools so that companies can handle the maintenance cost (Sneed, 2005). Moreover, automation can be considered as a problem derived from the standardization problem, since standardization and the formalization of the process are necessary requirements to provide tools to automate the process, which can be reused for several reengineering projects.

Figure 1. Horseshoe modernization model



Architecture-Driven Modernization

The software modernization paradigm, and particularly ADM as defined by the OMG, can be considered as a mechanism for software evolution, i.e., it makes it possible to modernize the legacy information systems and eradicates, or at least minimizes, the software erosion problem in legacy systems. According to (OMG, 2003b), ADM *"is the process of understanding and evolving existing software assets. ADM restores the value of existing applications"*.

ADM solves the problems of traditional reengineering since it carries out reengineering processes taking *model-driven* principles into account. However, ADM does not replace reengineering, but ADM improve it. The MDA (Model-Driven Architecture) standard proposed by OMG defines two main principles: (i) modeling all the artifacts as models at different abstraction levels; and (ii) establishing model transformations between them (Miller et al., 2003).

The horseshoe reengineering model has been adapted to ADM and it is known as the horseshoe modernization model (see Figure 1). There are three kinds of models in the horseshoe model (Miller et al., 2003):

- **Computation Independent Model (CIM)** is a view of the system from the computation independent viewpoint at a high abstraction level. A CIM does not show details of the system's structure. CIM models are sometimes called domain models and play the role of bridging the gap between the domain experts and experts in the system design and construction.
- **Platform Independent Model (PIM)** is a view of a system from the platform independent viewpoint at an intermediate abstraction level. A PIM has a specific degree of technological independence in order to be suitable for use with a number of different platforms of a similar type.
- **Platform Specific Model (PSM)** is a view of a system from the platform specific viewpoint at a low abstraction level. A PSM combines the specifications in the PIM with the details that specify how that system uses a particular type of platform or technology.

ADM solves the formalization problem since it represents all the artifacts involved in the reengineering process as models. Therefore,

ADM treats them homogenously, i.e., as models, meaning that ADM can establish model transformations between them. Those transformations are formalized by means of the QVT (Query / Views / Transformations) standard proposed by the OMG (OMG, 2008b). The QVT specification consists of two distinct but related languages: (i) *QVT-Operational* language, which is procedural in nature, and (ii) *QVT-Relations*, a declarative language. QVT makes it possible to define deterministic transformations between models at the same abstraction level or at a different level. As a consequence, the model transformations can be automated. Furthermore, the model-driven development principles make it possible to reuse the models involved in the ADM projects, since several PIM models can be generated from a CIM model, and in turn, several PSM models can be obtained for each PIM model. Therefore, the automation problem can be also solved due to the automated transformations together with the reuse of the models.

So far, model-driven development principles have been usually used in the forward engineering stage. Model-driven, generative techniques are used in that stage to obtain source code from different kinds of models, for instance UML models. Indeed, some researchers consider model-driven engineering only applicable for forward engineering. However, model-driven development principles can be applied to the reverse engineering and restructuring stages in an effective way.

In reverse engineering stage, the information recovered from software artifacts can be represented in models according to certain metamodels. In addition, restructuring and refactoring techniques can be applied on models. Thus, it consists of a transformation from the input model (as is) to obtain a target model (as be). Model-based restructuring has some advantages with respect to the traditional restructuring: (i) the model-based one allow researchers to define language- and platform-independent refactoring techniques; (ii) a restructuring transformation could be implemented as a model it-self, thus the transformation can be reused; (iii) the model-based refactoring makes it possible to define generic or domain-specific refactoring techniques in an easy manner; (iv) it improves the feature location since the traceability throughout corresponding models at different abstraction levels is better; (v) and so on.

ADM Standards

ADM not only adopts other existing standards like MDA and QVT, it has also spearheaded the development of a set of standards to address the different challenges that appear in the modernization of legacy information systems (OMG, 2009b).

In June 2003, the OMG formed a Task Force for modeling software artifacts in the context of legacy systems. Initially, the group was called the Legacy Transformation Task Force and was then renamed the Architecture-Driven Modernization Task Force (ADMTF). In July 2003, the ADMTF issued a software modernization whitepaper (OMG, 2003b). In November 2003, the ADMTF issued the request-for-proposal of the Knowledge Discovery Metamodel (KDM) specification. KDM aims to be an initial metamodel that allows modernization tools to exchange application metadata across different applications, languages, platforms and environments. The objective of this initial metamodel was to provide a comprehensive view of the application structure and data, but does not represent software below the procedure level. The request-for-proposal stated that the KDM metamodel:

- represents artifacts of legacy software as entities, relationships and attributes
- includes external artifacts with which software artifacts interact
- supports a variety of platforms and languages

- consists of a platform and language independent core, with extensions where needed
- defines a unified terminology for legacy software artifacts
- describes the physical and logical structure of legacy systems
- can aggregate or modify, i.e. refactor, the physical system structure
- facilitates tracing artifacts from logical structure back to physical structure
- represents behavioral artifacts down to, but not below, the procedural level.

In May 2004, six organizations responded to the request-for-proposal. However, throughout 2004 and 2005 more than 30 organizations from 5 different countries have collaborated in the development and review of the KDM standard. In May 2006, KDM was adopted by the OMG and moved into the finalization stage in the adoption process. In March of 2007, the OMG presented the recommended specification of KDM 1.0. In April 2007, the OMG started ongoing maintenance of the KDM specification. In January 2009, the recommended specification of KDM 1.1 became available in the OMG (OMG, 2009a), and in turn, the OMG started the revision of this version. Recently, in March 2009, KDM was recognized as a draft international standard, specifically ISO/IEC 19506 (ISO/IEC, 2009).

KDM is the first fulfilled standard and is the cornerstone of the set of standards proposed by the ADMTF of the OMG. However, the ADMTF is defining the rest of the standards planned around the KDM, although some of these standards are still in the approval or development stage. The set of ADM specifications is (OMG, 2009b):

- **Abstract Syntax Tree Metamodel (ASTM)** is a specification built under KDM to represent software below the procedural level by means of abstract syntax tree models. The ASTM and the KDM are two complementary modeling specifications to fully represent applications and facilitate the exchange of granular metadata across multiple languages. In 2009 the ADMTF proposed the first specification of ASTM 1.0.
- **Software Metrics Metamodel (SMM)** defines a metamodel for representing measurement information related to software, its operation, and its design. The specification is an extensible metamodel for exchanging software-related measurement information concerning legacy software artifacts: designs, implementations, or operations. Also in 2009, the ADMTF proposed the first specification of SMM 1.0.
- **ADM Pattern Recognition specification** aims to be a standard to facilitate the examination of structural metadata with the intent of deriving patterns about legacy systems. These patterns can be used to determine refactoring and transformation requirements and opportunities that could be applied to one or more systems. This specification was issued in 2009.
- **ADM Visualization specification** focuses on the different ways to show application metadata stored within the KDM models. In 2009 there was no target date for this proposal.
- **ADM Refactoring specification** seeks to define ways in which the KDM specification can be used to refactor applications by means of structuring, rationalizing, modularizing, and so on. In 2009 there was no target date for this proposal.
- **ADM Transformation specification** seeks to define mappings between KDM, ASTM and target models. Also, in 2009 there was no target date for this proposal.

Software Evolution through ADM

Software evolution is a kind of software maintenance which is also called evolutionary maintenance. In general, the maintenance process can perform four categories of modifications in the existing software (ISO/IEC, 2006):

- **Corrective maintenance** modifies a software product performed after delivery in order to correct discovered problems.
- **Preventive maintenance** modifies a software product after delivery to detect and correct latent faults in the software product before they become effective faults.
- **Adaptive maintenance** modifies a software product performed after delivery to keep a software product usable in a changed or changing environment;
- **Perfective maintenance** modifies a software product after delivery in order to improve the performance or the maintainability.

Evolutionary maintenance is a specific maintenance process that focuses on adaptive and perfective modifications. Indeed, (Mens et al., 2008) states that a legacy system must be evolved when it *"operates in or addresses a problem or activity of the real world. As such, changes in the real world will affect the legacy system and require adaptations to it"*.

Software evolution can be considered as a process to enhance legacy information systems in order to deal with software erosion problems. Nevertheless, the software erosion phenomenon is due to the maintenance process itself. Thus, according to the software entropy law, evolutionary maintenance could trigger more software erosion problems over the long term (Jacobson et al., 1992).

"The software entropy law states that a closed system's disorder cannot be reduced, it can only remain unchanged or increase. A measure of this disorder is entropy. This law also seems plausible for software systems; as a system is modified, its disorder, or entropy, always increases. This is known as software entropy". (Jacobson et al., 1992)

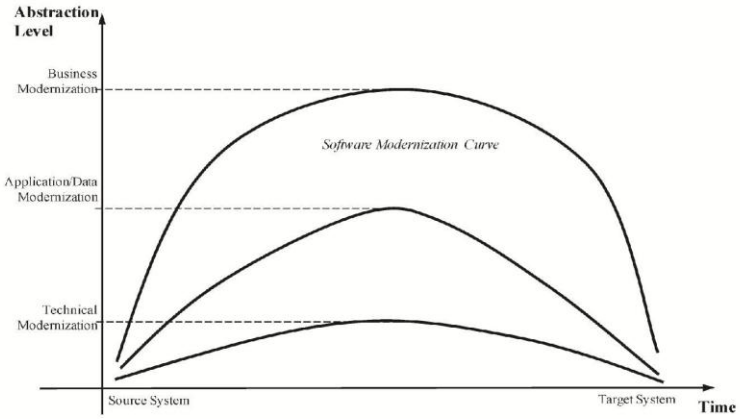
Therefore, evolutionary maintenance should generate new improved versions of legacy systems while minimizing the negative effects of the maintenance. Software modernization, and in particular ADM itself, can be seen as a kind of evolutionary maintenance, which is based on the reengineering process and follows a model-driven approach. Due to the fact that ADM advocates carrying out model-driven processes, automation and reuse (based on abstract models) are ensured. Thus, the outcomes of the ADM-based processes are not only the improvement or evolution of a specific legacy system, but also that the legacy system is represented through a set of models at different abstraction levels that can be reused in future maintenance processes. Therefore, the software entropy problem is minimized with software evolution by means of ADM.

According to the horseshoe model, the ADM-based process can be categorized into three kinds of modernization processes (Khusidman et al., 2007). These depend on the abstraction level reached in the reverse engineering stage in the horseshoe model, and thus each kind of modernization process defines a specific modernization curve (see *Figure 2*).

Depending on the abstraction level reached (see *Figure 2*), the knowledge and models available to the restructuring stage may be different. Usually, a higher abstraction level implies a greater amount of knowledge and rich information that provides more restructuring possibilities to the modernization process. The three kinds of modernization processes are the following:

- **Technical Modernization.** This kind of modernization considers the lowest ab-

Figure 2. Horseshoe modernization model



straction level and is historically the most commonly applied to legacy systems. A company carries out a technical modernization project when it wants to deal with platform or language obsolescence, new technical opportunities, conformance to standards, system efficiency, system usability or other similar modernization factors. This type is sometimes not considered strictly as a modernization process since it only focuses on corrective and preventive modifications, but in any case it addresses adaptive or perfective modifications according to the modernization definition.

- **Application/Data Modernization.** This kind of modernization considers an intermediate abstraction level since it focuses on restructuring a legacy system at the level of application and data design to obtain the target system. This kind of modernization is driven by several modernization factors such as improving the system reusability, reducing the delocalized system logic or system complexity, and applying design patterns. There is a fine line between this kind of modernization and the previous one, but that line is crossed when there is some impact on the system design level.
- **Business Modernization.** This kind of modernization increases the abstraction level to the maximum. Thus, the restructuring stage is done at the level of business architecture, i.e., the business rules and processes that govern a legacy system in the company. This kind of modernization incorporates, apart from technical models and application/data models, business semantic models that are a key asset (i) to preserve the business knowledge embedded in legacy systems; and (ii) to align the business requirements of the company with the future target systems.

Business modernization is probably the most important kind of modernization since changes in business processes are one of the most important software modernization decision criteria according to (Koskinen et al., 2005). However, not many modernization attempts have achieved the business abstraction level in the reverse engineering stage so far. This could be due to the fact that the mapping paradigms between business and IT lack the standardization to modernize legacy systems.

For this reason, the standardization effort by the ADMTF is so important. As a consequence, the ADM standards make it possible to deal with this kind of modernization processes at this time.

Modernization Scenarios

Each kind of modernization process makes it possible to modernize certain legacy systems under specific modernization scenarios. A modernization scenario provides templates that define project objectives, tasks and their related deliverables, the role of each ADM standard, and so on.

"A modernization scenario can be defined as an initiative, such as the ongoing task of portfolio management, a project, such as migrating from one platform to another, or a series of project phases (i.e., a super project) as would be the case when consolidating, redesigning and redeploying an application in a model-driven architecture". (OMG, 2006b)

According to (OMG, 2006b), there are thirteen different modernization scenarios that can be applied to legacy systems:

1. **Application Portfolio Management.** Companies have several applications and information systems which are grouped together under the name application portfolio. The application portfolio should be managed as business assets, meaning that this scenario aims to capture and expose a repository of cross-systems and cross-platform application metadata in a company's systems. This is due to the fact that legacy systems are usually poorly documented and this scenario addresses this shortcoming. This scenario is applied within application/data or business modernization processes.
2. **Application Improvement.** This scenario aims to improve the robustness, integrity, quality, consistency and/or performance of applications, but it does not involve an architecture transformation effort. The scenario can be thought of as a super scenario that consists of several sub-scenarios such as the correction of system flaws, source code restructuring, source code rationalization and other refactoring activities. This scenario is only applied within technical modernization processes.
3. **Language-to-Language Conversion.** This scenario deals with the conversion of one or more information systems from one programming language to another. The scenario can be motivated by the obsolescence of a language, or by the existence of a requirement to enhance a functionality not supported in the current language, and so on. However, this scenario does not involve a redesign of the system functionality. Thus, it is applied within technical modernization processes.
4. **Platform Migration.** This scenario moves systems from one platform to another because of the platform's obsolescence or to standardize the system to an organizational standard. The scenario is usually carried out together with a language-to-language conversion. This scenario is also applied throughout technical modernization processes.
5. **Non-Invasive Application Integration.** This scenario is triggered when there is a need to bring legacy user interfaces to end users by replacing the legacy front-ends with other front-ends like Web-based interfaces. Thus, the users gain value through replacing legacy front-ends while leaving the core functionality of the system untouched. This scenario is only carried out within technical modernization processes.
6. **Service Oriented Architecture Transformation.** This scenario takes the legacy systems that usually embed their functionality monolithically and segregates that functionality into services according

to SOA principles. The scenario not only modifies the front-end interfaces, but it also requires the redesign and componentization of the whole legacy system. In the scenario, the KDM standard identifies and tracks relationships between the physical system, program functionality, data usage and user interfaces. This scenario needs to be applied within application/data modernization processes.

7. **Data Architecture Migration.** This scenario moves one or more data structures to another, for instance, moving from a non-relational file or database to the relational data architecture. ADM also facilitates this scenario because the KDM metamodel assists with tracking and refactoring artifacts such as program-based data definitions, data access logic, and so on. The scenario must be applied within application/data modernization processes.
8. **System and Data Architecture Consolidation.** This scenario is motivated by the need to build a single system from multiple standalone systems that perform the same basic functions. The scenario aims to eradicate the redundant systems and data structures, inconsistencies, redundant business processes, and so on, in order to minimize excessive maintenance workloads. It can be combined with other scenarios such as model-driven transformation (see number 12), language change or platform migration. This scenario is applied throughout application/data modernization processes.
9. **Data Warehouse Deployment.** This scenario builds a common repository or a data warehouse of business data as well as the different ways to access this data. The KDM standard facilitates data analysis and data capture from legacy systems. In addition, the Common Warehouse Model (CWM) (OMG, 2003a), another OMG standard but outside the ADMTF, can be used together with KDM to represent data from different data sources in order to build and maintain the data warehouse. This scenario is usually carried out within application/data modernization processes, but it can be applied within business modernization processes as well.
10. **Application Package Selection and Deployment.** This scenario is aimed at third party application components. It defines how legacy systems are to be retired, integrated or retooled to work with a specific component. Furthermore, it helps to determine which portions of a component need to be implemented, integrated, discarded or updated. This scenario is exclusively applied within application/data modernization processes.
11. **Reusable Software Assets / Component Reuse.** This modernization scenario helps to identify, capture, streamline and prepare functionality and information assets of legacy systems for reuse. The ADM standards facilitate the identification of specific software artifacts based on data usage, transaction access, and other criteria. This scenario is also applied within application/data modernization processes.
12. **Model-Driven Architecture (MDA) Transformation.** This scenario converts a non-model-driven environment into a model-driven environment. Moving to MDA requires transforming legacy systems into a set of models that can be used to generate replacement systems. This scenario is applied within application/data modernization processes.
13. **Software Assurance.** This scenario aims to measure trustworthiness with regard to established business and security objectives. The KDM standard makes it possible to verify and measure software attributes across a variety of platforms and languages. Also, SMM can be used to define, evaluate and visualize different software metrics about legacy systems. This scenario can be

applied within application/data or technical modernization processes.

Toward Software Archeology

The reverse engineering stage is probably the most important stage in the horseshoe modernization process. This is due to the fact that this activity conditions the abstraction level achieved in each kind of modernization process and therefore the resources and possibilities to restructure the legacy systems. The reverse engineering stage is the key activity especially when a business modernization process is being carried out. In the third kind of modernization process, the reverse engineering stage must obtain business process models, business rules and any other business semantics from the legacy systems. However, there is a large conceptual gap between business processes and legacy systems that needs to be gradually reduced. Therefore, specific business knowledge must be extracted, although in many cases, it must be also inferred or deduced from previous knowledge.

Therefore, the reverse engineering stage can also be seen as a software archeology process in modernization processes. Real archeologists investigate several artifacts and situations, trying to understand what they are looking at, i.e., they must understand the cultural and civilizing forces that produced those artifacts. In the same way, a software archeologist analyzes different legacy artifacts such as source code, databases and user interfaces and then tries *"to understand what they were thinking to understand how and why they wrote the code the way they did"* (Hunt et al., 2002). Therefore, the software archeology process consists of analyzing different software artifacts by means of reverse engineering techniques and tools in order to obtain very abstract models that depict not only the legacy systems, but also the company and/or the company operation supported by this system, e.g., business process models.

ADM facilitates the software archeology process by means of KDM, since this standard makes it possible to represent all software artifacts involved in a certain legacy system in an integrated and standardized way. KDM can be compared with the UML standard (OMG, 2009c): UML is used to generate new code in a top-down manner; but in contrast, a process involving KDM starts from the existing code and builds a higher level model in a bottom-up manner (Moyer, 2009).

KDM Metamodel

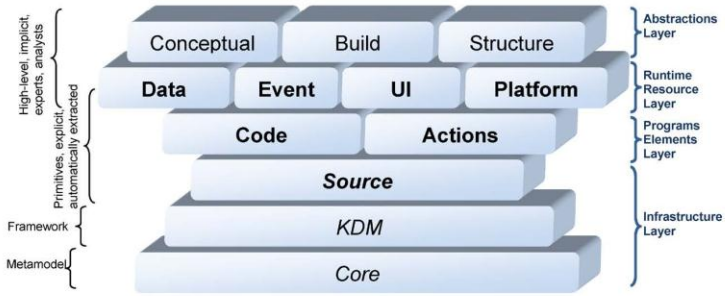
The goal of the KDM specification (OMG, 2009a) is to represent legacy information systems as a whole and not just their source code (Khusidman, 2008). The KDM metamodel provides a comprehensive high-level view of the behavior, structure and data of systems, but it does not represent the procedural models of the systems.

The KDM metamodel is divided into layers representing both physical and logical software artifacts of information systems at several abstraction levels (OMG, 2008a). It separates knowledge about legacy information systems into various orthogonal concerns that are known in software engineering as architecture views (see Figure 3).

The KDM metamodel consists of four abstraction layers, each based on a previous layer. Furthermore, each layer is organized into packages that define a set of metamodel elements whose purpose is to represent a specific independent facet of knowledge related to legacy systems:

- **Infrastructure Layer** defines a small set of concepts used systematically throughout the entire KDM specification. It has three packages: Core, KDM and Source. The Core and KDM packages define the common metamodel elements that constitute the infrastructure for other packages. The Source package enumerates the artifacts of the legacy system and defines traceability links between the KDM elements and their original representation in the legacy source code.

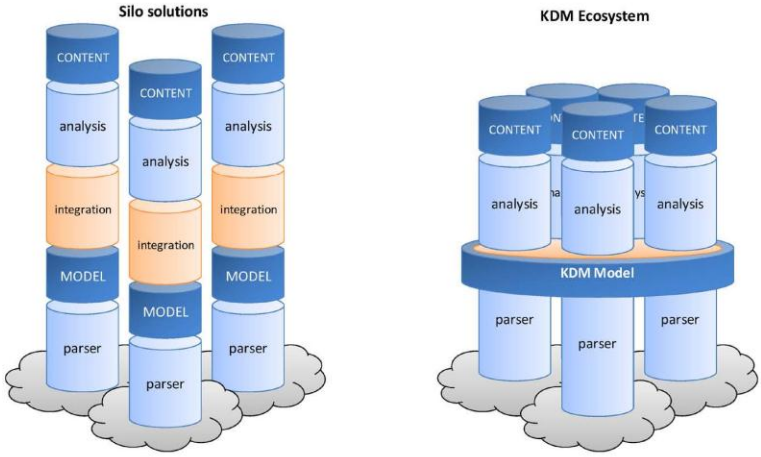
Figure 3. Layers, packages and concerns in KDM (Adapted from (OMG, 2008a)).



- **Program Elements Layer** aims to provide a language-independent intermediate representation for various constructs determined by common programming languages. There are only two packages in this layer: Code and Action. The Code package represents the named items from the source code and several structural relationships between them and the Action package focuses on behavior descriptions and control- and data-flow relationships determined by them.
- **Runtime Resource Layer** enables the representation of high-value knowledge about legacy systems and their operating environment, i.e., this layer focuses on those things that are not contained within the code itself. It has four packages: Data, Event, UI and Platform. The Data package defines the persistent data aspects of a legacy system; the Event package defines a common concept related to event-driven programming; the UI package represents the user interface aspects; and finally, the Platform package defines artifacts related to the run-time platform and environment of a legacy system.
- **Abstraction Layer** defines a set of metamodel elements whose purpose is to represent domain-specific knowledge as well as to provide a business-overview of legacy information systems. This layer has three packages: Conceptual, Structure and Build. The Conceptual package represents the domain-specific elements of a legacy system; the Structure package represents the architectural components of legacy systems like subsystems, layers, packages and so on; and the Build package defines the artifacts related to engineering legacy systems such as roles, development activities or deliverables.

Throughout the software archeology process, the Infrastructure layer is not used to represent any specific model since the elements of this layer are used by other layers. The following KDM layers, Program Elements layer and Runtime Resource layer are used for modeling explicit knowledge of the legacy systems. Therefore the models built according to the packages of these layers can be obtained automatically through specific tools. For instance, imagine a tool that analyzes a database schema and directly represents a KDM data model from the recovered information. The last layer, the Abstraction layer, is used when the software archeologist finds himself full-on interpreting intent (Moyer, 2009). That is, this layer focuses on representing the implicit knowledge of the legacy systems. For this purpose, the models built according to this layer are obtained by means of the analysis of the explicit knowledge of the previous layers as well as the help of domain experts.

Figure 4. Software archeology tools: silo solutions (left) and KDM ecosystem (right)



The KDM Ecosystem

Legacy information systems are becoming more and more complex, and the development and management of these systems already require an important effort. However, the software archeology processes as a part of an ADM-based process involve, perhaps, even greater effort in order to modernize or simply understand the complex legacy systems. Therefore, the software archeology processes (or most parts of them) must be automated by means of reverse engineering tools.

In this respect, the KDM standard enables the automation of the software archeology processes since it defines the items that a reverse engineering tool should discover and a software analysis tool can use; the KDM standard, then, has been designed to enable knowledge-based integration between tools.

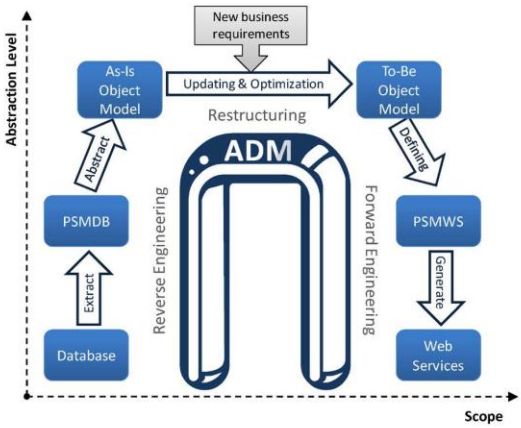
The KDM standard changes the way in which reverse engineering tools are built and used. Traditional reverse engineering tools have been built as silos where each tool recovers and analyzes different proprietary content in a single silo (see Figure 4, left). For instance, suppose that we are carrying out an archeology process of a legacy enterprise application. We would use a reverse engineering tool for the source code and another tool for the legacy database. As a consequence, at the end of the process, we would have two proprietary and independent models, a source code model and a database model, which must be also analyzed independently.

The KDM standard also makes it possible to build reverse engineering tools in a KDM ecosystem (see Figure 4, right). Here, the reverse engineering tools recover different knowledge related to different artifacts, but this knowledge is represented and managed in an integrated and standardized way through KDM. Thus, the software analysis tools can analyze the KDM repository and generate the new knowledge. Furthermore, in the future more software analysis tools can be plugged in to the KDM models homogeneously, to generate ever more valuable knowledge.

A MODERNIZATION CASE STUDY

This section shows a case study that applies an ADM-based process for modernizing legacy

Figure 5. An ADM-based process form modernizing relational databases



databases toward SOA environments. The case study consists on a real modernization project (Pérez-Castillo et al., 2009b) which was jointly carried out by the University of Castilla-La Mancha and Indra Software Labs (a multinational software development company) in the context of the "Catedra Indra" (CI hereinafter), a R+D centre located in Spain and devoted to carry out research projects in a close cooperation between industry and university.

Context of the Case Study

CI needed to develop a corporate website in order to support all the information related to the cooperation of industry & university. For instance, the site should contain information about conferences, lectures, courses, offered grants, events, awards, published papers, journals and so on. Thus, the website could be used by academics, researchers, PhD candidates and students.

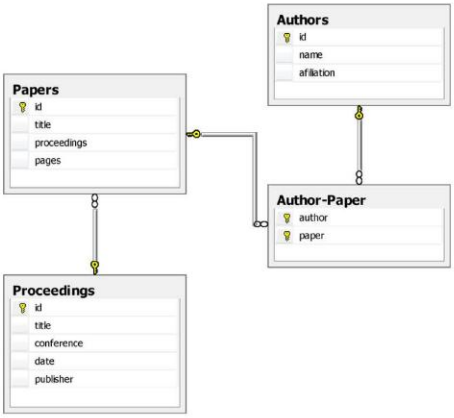
The CI website must be built using a standard Web architecture based on the Microsoft.NET platform, and it uses Microsoft SQL Server 2000 as RDBMS (Relational Data Base Management System).

Moreover, CI was using at that moment an existing database created long time ago, which was used to register the some information related to the CI. However, the data stored in database was not being managed by any application. In this context, it is possible to find the suitable conditions for implementing a modernization project, since the CI's database is considered as a legacy system which can be exposed as web services in order to integrate in a SOA environment. Therefore, a modernization process based on ADM was proposed to obtain a set of web services from the legacy database to manage the access to the data, and then integrate the web services into the CI website.

The Proposed Modernization Process

The process aims to establish guidelines to allow the generation of Web services from relational databases through a set of model transformations. Figure 5 represents the proposed modernization process. Firstly, a PSM model, according to SQL-92 meta-model (ISO/IEC, 1992), is obtained from the legacy database through reverse engineering. Secondly, the PSM is transformed to PIM model

Figure 6. Fragment of the legacy database schema modernized in the case study



which raises the abstraction level of the system. This PIM model is represented in terms of UML2 meta-model (OMG, 2007b). Thirdly, the restructuring stage can modify the object model in order to add new business requirements. Fourthly, the process generates through forward engineering a certain PSM model that depicts the web services at higher abstraction level. Finally, the source code of the web services is generated from the PSM model, thus the abstraction level is reduced again.

The modernization process (see Figure 5) is explained in the following subsections in detail. In addition, an example will be developed to clarify each stage in the process. The example considers only a small fragment of the legacy database of CI (see Figure 6). The database fragment consists of four tables: *Proceedings*, *Papers*, *Authors* and *Author-Paper*. The proceedings have several papers, which are written by several authors in turn. The table *Author-Paper* relates each paper with the authors.

Reverse Engineering Stage

The first stage is the database reverse engineering, which aims to recover the relational database design and create a PSM model that represents the input database. The metadata recovered from the database is represented in a PSM model according to the SQL-92 metamodel (see Figure 7).

The metadata needed to build the database schema model can be taken through *INFORMATION_SCHEMA* (Melton et al., 1993). This is a standardized mechanism from SQL-92 standard that identifies a set of views. These views return metadata in a standardized way and in a RDBMS-independent manner.

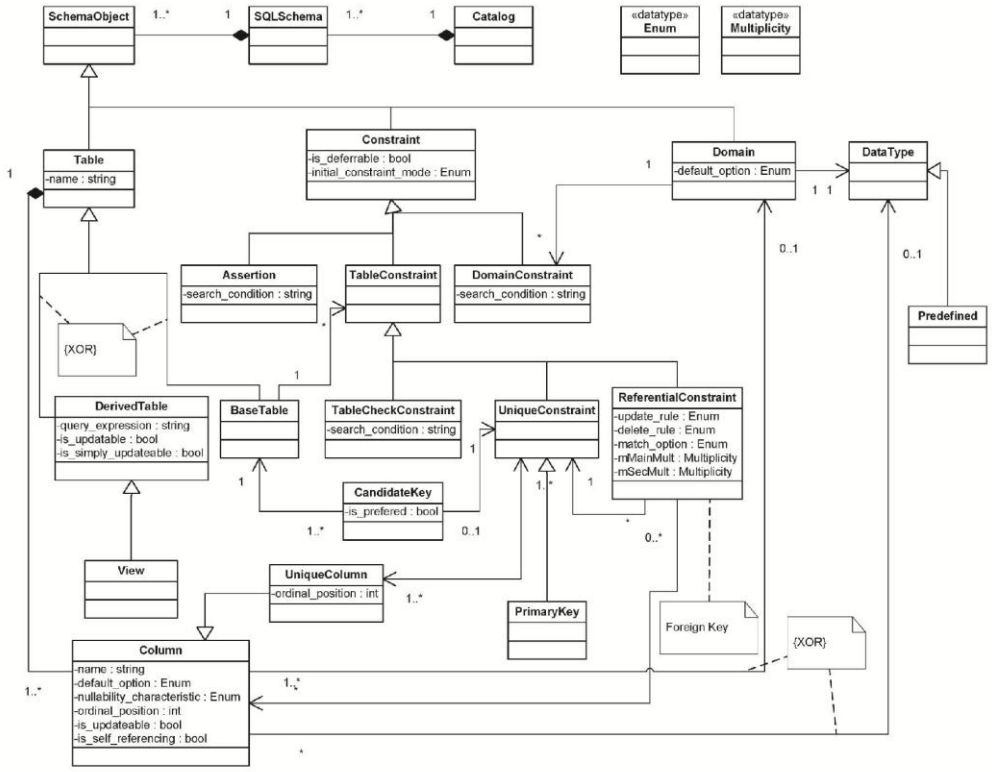
Moreover, the built model can be made persistent through XMI (*XML Metadata Interchange*) (Grose et al., 2001), which defines the manipulation, persistence and interchange of models through XML. In the example, the database model is recovered and written in XMI format (see Figure 8). In this model, there is an element for each table and constraint in database schema according to the proposed metamodel. There are four instances of the *BaseTable* metaclass to represent the four tables. Also, the different constrains are modelled as instances of the *PrimaryKey* and *ReferentialConstraint* metaclass (see Figure 8).

After database design is recovered, the outgoing PSM model is transformed into a PIM model that represents the database information in a platform-independent manner. Thus, in this step an object model is generated, which will be afterward the basis to generate Web Services in following stages.

The object model is represented according to UML2 metamodel (OMG, 2007b) and the transformation is defined by means of QVT. Figure 9 shows a QVT fragment of the transformation. This fragment presents two relations: the first one transforms each database schema into a code package, and the second one generates instances of the *Class* metaclass for each instance of the *Table* metaclass of the PSM model.

The object model obtained in the example consists of four objects, one object for each table in database model: *Proceedings*, *Papers*, *Authors*, *Author-Paper*. These classes conforms the so-

Figure 7. SQL-92 meta-model



called *business layer* according to the three-tier architecture (see Figure 10), which contains all business knowledge of the future web services. The remaining layers are addressed in following stages of the modernization process.

Restructuring Stage

The restructuring stage is in charge of transform the PIM model into another improved PIM model at the same abstraction level. This stage adds the auxiliary objects to support the persistence layer of the object model (see Figure 10).

In addition, the restructuring stage discovers the potential services from the information recovered from the database. For this purpose, the modernization process defines a set of patterns, which are sought in the recovered database schema. In turn, each pattern defines a template to create well known services. Figure 11 details the patterns as well as the services that can be derived from each pattern. On the one hand, there are simple services involving only a single table. These services are directly obtained from database scheme and matched with *CRUD* operations (*Create / Read / Update / Delete*) as well as *getters & setters* methods for handling the different columns on each table. And on the other hand, advanced services involve several tables of schema (see Figure 11). In this case, services may be directly obtained from views, or on the contrary, services can be obtained from the following patterns that

Figure 8. Database model (in XMI representation) obtained in the modernization case study


```
  <?xml version="1.0" encoding="utf-8" ?>
- <xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI">
  - <xmi:Documentation>
      <exporter>PRECISO</exporter>
      <exporterVersion>0.5</exporterVersion>
    </xmi:Documentation>
  - <Schema name="Conferences">
    - <SchemaObjects>
      - <Tables xmi:id="_0">
        + <BaseTable name="Papers" xmi:id="_1.0">
        - <BaseTable name="Proceedings" xmi:id="_1.1">
          - <Columns>
            + <Column name="id" xmi:id="_1.1.0" ordinal_position="1"
                is_nullable="NO">
            + <Column name="title" xmi:id="_1.1.1" ordinal_position="2"
                is_nullable="NO">
            + <Column name="conference" xmi:id="_1.1.2"
                ordinal_position="3" is_nullable="NO">
            + <Column name="date" xmi:id="_1.1.3" ordinal_position="4"
                is_nullable="NO">
            + <Column name="publisher" xmi:id="_1.1.4"
                ordinal_position="5" is_nullable="NO">
            </Columns>
          </BaseTable>
        + <BaseTable name="Authors" xmi:id="_1.2">
        + <BaseTable name="AuthorPaper" xmi:id="_1.3">
        + <BaseTable name="sysdiagrams" xmi:id="_1.4">
        </Tables>
      - <Constraints xmi:id="_1">
          <Assertions />
        - <TableConstraints>
          - <UniqueConstraints>
            + <UniqueConstraint name="UK_principal_name" xmi:id="_2.8">
            </UniqueConstraints>
          + <PrimaryKeyConstraints>
          - <ReferentialConstraints>
            + <ReferentialConstraint name="FK_Paper_Proceedings"
                xmi:id="_2.4" update_rule="NO ACTION" delete_rule="NO
                ACTION" match_option="SIMPLE"
                xmlns="FK_Paper_Proceedings">
            + <ReferentialConstraint name="FK_Author-Paper_Author"
                xmi:id="_2.5" update_rule="NO ACTION" delete_rule="NO
                ACTION" match_option="SIMPLE" xmlns="FK_Author-
                Paper_Author">
            + <ReferentialConstraint name="FK_Author-Paper_Paper"
                xmi:id="_2.6" update_rule="NO ACTION" delete_rule="NO
                ACTION" match_option="SIMPLE" xmlns="FK_Author-
                Paper_Paper">
            </ReferentialConstraints>
            <TableCheckConstraints />
          </TableConstraints>
          <DomainConstraints />
        </Constraints>
        <Domains />
      </SchemaObjects>
    </Schema>
  </xmi:XMI>
```


are recognized in the relational database scheme: (1) *referenced table*, when there is a foreign key among two tables; (2) *combined table*, when there are two or more foreign keys from one table to other; and finally (3) *observed table*, which searches two or more foreign keys in different tables to a table.

According to the example, the process adds four objects to support the persistence layer: *Procedings_persistent*, *Papers_persistent*, *Author_persistent* and *AuthorPaper_persistent* (see Figure 10). In addition, several services are discovered by means of the proposed patterns. Let us pay attention to the table *Author-Paper*, which

Figure 9. QVT fragment to transform a database model (PSM) into an object model (PIM)

```
transformation rdbms2uml(rdbms:SimpleRDBMS, uml:SimpleUML) {
   -- map each schema to a package
   top relation SchemaToPackage {
     pn : String;
     checkonly domain rdbms s : RdbmsSchema {
       rdbmsName = pn
     };
     enforce domain uml p : UmlPackage {
      umlName = pn
     };
   }
   -- map each table to a class
   top relation TableToClass {
     cn : String;
     checkonly domain uml t : RdbmsTable {
       rdbmsSchema = s : RdbmsSchema {},
       rdbmsName = cn
     };
     enforce domain rdbms c : UmlClass {
       umlNamespace = p : UmlPackage {},
       umlName = cn
     };
     when {
       SchemaToPackage(s, p);
     }
     where {
       -- Call to ColumnToAttribute relation for each column in each class
       t.rdbmsColumn->forAll(cl:RdbmsColumn | cl.oclIsUndefined()
         implies ColumnToAttribute(t, c, cl));
     }
   }
   ...
}
```

Figure 10. Object model obtained in the modernization case study

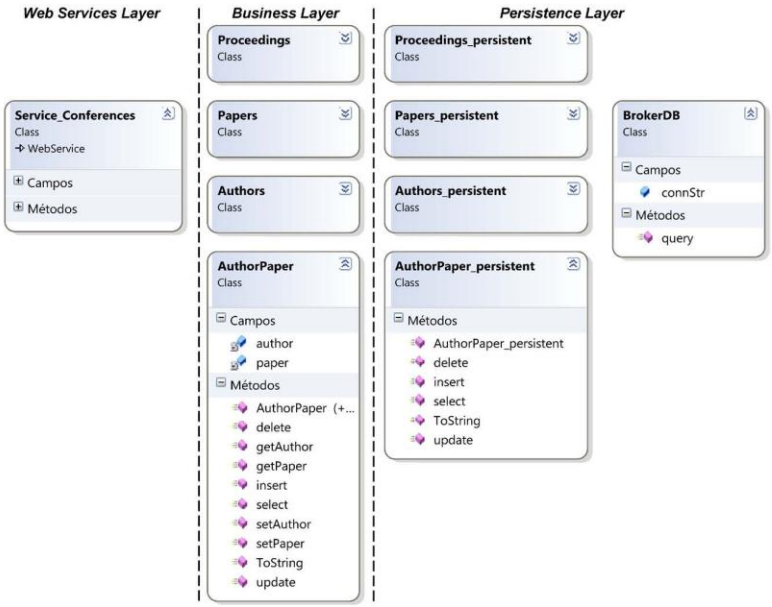
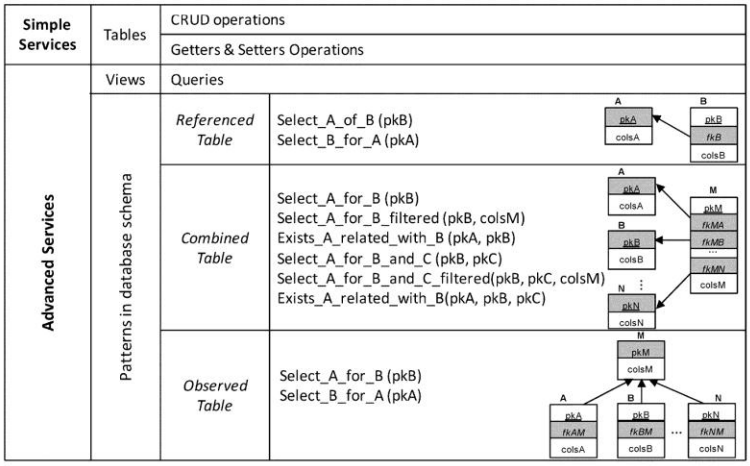


Figure 11. Patterns and service templates used in the modernization case study



have two foreign keys to the tables *Authors* and *Papers* (see Figure 6). The *'Combined Tables'* pattern is recognized in that structure of the database schema, therefore predefined services could be built from the detected pattern instance (see Figure 12). The service templates to select tuples filtering data are not instantiated in this example because the table *Author-Paper* has not additional columns a part from the columns involved in the primary key (see Figure 6).

Forward Engineering Stage

The third stage aims to obtain by means of forward engineering a set of web services that manage the access to the data stored in the input database. Firstly, the last PIM model (transformed after restructuring stage) is transformed into a PSM model that represent the service interfaces of the future web services. This model is represented according to the WSDL metamodel (Web Services Description Language) (W3C, 2007).

In the example, it focuses again on the instance of the *'Combined Tables'* pattern that was discovered, and in particular it shows the results for the service *'selectAuthors_for_Papers (int id_paper)'*. Figure 13 shows the WSDL segment generated for this service by means of the process from Service Discovery and Object Model Transformation tasks.

Finally, the last step is the generation of the source code to support: (i) the object model obtained in previous stages, as well as (ii) the implementation of the web services from the WSDL model.

Figure 12. Services obtained from an instance of 'Combined Tables' pattern

```
Author selectAuthors_for_Papers(int id_paper)
Boolean existsAuthors_rel_Papers(int id_author, int id_paper)
Paper selectPapers_for_Authors(int id_author)
Boolean existsPapers_rel_Authors(int id_paper, int id_author)
```

Figure 13. The WSDL segment generated to support a service in the modernization case study

```
  <?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:tns="http://Conferences.org/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    targetNamespace="http://Conferences.org/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Web
      service for testing access to database by means of Conferences
      objects.</wsdl:documentation>
  - <wsdl:types>
    - <s:schema elementFormDefault="qualified"
        targetNamespace="http://Conferences.org/">
      - <s:element name="selectAuthors_for_Papers">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="paper"
                type="s:int" />
            </s:sequence>
          </s:complexType>
        </s:element>
      + <s:element name="selectAuthors_for_PapersResponse">
      </s:schema>
    </wsdl:types>
  + <wsdl:message name="selectAuthors_for_PapersSoapIn">
  - <wsdl:message name="selectAuthors_for_PapersSoapOut">
      <wsdl:part name="parameters"
        element="tns:selectAuthors_for_PapersResponse" />
    </wsdl:message>
  - <wsdl:portType name="Service_ConferencesSoap">
    - <wsdl:operation name="selectAuthors_for_Papers">
        <wsdl:documentation
          xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Return the
          object list correspond to the pattern "COMBINED TABLE" to
          relationship "selectAuthors_for_Papers"</wsdl:documentation>
        <wsdl:input message="tns:selectAuthors_for_PapersSoapIn" />
        <wsdl:output message="tns:selectAuthors_for_PapersSoapOut" />
      </wsdl:operation>
    </wsdl:portType>
  + <wsdl:binding name="Service_ConferencesSoap"
      type="tns:Service_ConferencesSoap">
  + <wsdl:binding name="Service_ConferencesSoap12"
      type="tns:Service_ConferencesSoap">
  + <wsdl:service name="Service_Conferences">
  </wsdl:definitions>
```

In the example, the classes shown in Figure 10 are written in physical files according to a certain programming language that supports Web Services technology as *Java*, *C#*, and so on. In this case study, the web services are written in *C#*. Finally, the obtained web services are deployed moving them to production stage they are integrated into the CI website. Thus, the web services become in fully operational services that wrap the legacy database and provides a modernized data access in a SOA environment.

Obtained Results

In order to support the modernization process, an ad hoc tool was developed to carry out the modernization of the legacy database. The tool made a selective publication and deployment of some services from the entire set of web services discovered from the database. Figure 14 summarizes the performance of the services used in the CI website. This table considers (1) the different types of services according to the set of patterns; (2) services generated for each type; (3) the number of services (for each type) included in the CI website; and (4) the percentage of services included.

Figure 14. Performance of services published in CI website

Kind of Service	Candidate Services	Published Services	Performance
insert	13	11	85%
update	13	11	85%
delete	13	11	85%
select	13	12	92%
setters	56	0	0%
getters	56	0	0%
show	13	0	0%
views	4	4	100%
referenced tables	24	12	50%
combined tables	32	8	25%
observed tables	8	4	50%
TOTAL	245	73	30%

The percentage of services that was published to support the functionalities of the CI website was 30%. This percentage included 73 services on a total of 245 candidate services. In addition, the development staff noticed that the non-selected web services would be very useful for future developments. Since these non considered services were identified and collected, it would be easy to deploy and integrate them into the CI website for the implementation of additional features. The tool keeps information of the modernization project, that is, the object model, configuration of the generated services, services deployed and services available. Thus, we only need to load the project of the current case study to deploy new web services to fulfill new information requirements.

The development staff took advantages from the availability of the required information to improve the development process: since all the required information was available as services, the staff could work with real data when developing the website. The staff could put all their effort into the development of the web interface. Furthermore, since the required information was available from the first, all the features of the CI site could be tested with the real information of the database. It allowed the staff to accelerate the testing process, because web developers could build the necessary web interfaces which in turn will use the aforementioned web services, and will properly display the information.

ADM Success Stories

In spite of the fact that ADM is a relatively new approach, it has acquired great relevance not only within the academic community but also within the industrial community. Currently, there are several international projects as well as tools that address modernization challenges.

Modernization Projects

Several meaningful and international projects such as MOMOCS (MOMOCS, 2008) and ModelWare (ModelWare, 2006) aim to lead ADM industrialization and also ensure its successful adoption by the industry. Both projects are funded by the European Commission under the Information Society Technologies in the 6th Framework Program. These projects combine innovations in modeling technologies, engineering processes and methodologies, tool development, standardization, experimentations and change management.

The MOMOCS project studies how complex legacy systems can be modernized with a focus on some of their specific software portions, with the goal of *"keeping up with a very fast changing business and technical environment taking human beings as the centre of the interaction"* (MO-

MOCS, 2008). The MOMOCS project defines a modernization methodology that is applied to complex systems and makes it possible to reengineer software architectures, data heterogeneities and processes in order to make system behavior more predictable in terms of performance, stability and maintainability.

The ModelWare project has developed a complete infrastructure required for the large-scale deployment of model-driven development strategies, validating this infrastructure in several business domains including the modernization of legacy systems (ModelWare, 2006). While the ModelWare project focuses on most parts of model-driven development, the MOMOCS project particularly addresses the activities and techniques of the software modernization approach.

Moreover, in 2009 the OMG organized an *Architecture Driven Modernization Case Studies Workshop*. Some relevant works were presented in this event, which consist of real world case studies involving the automated transformation of information systems from legacy source code into modernized languages and platforms. For instance, *DelaPeyronnie et al.* (DelaPeyronnie et al., 2010) present an ADM-based project for modernizing the EATMS system, an air traffic management system used at 280 airports worldwide. This project was carried out in a platform migration scenario, since the main objective was the transformation of the legacy system from the *Ada* language into the high-performance, real-time *Java* language. Another successful case study was the modernization project presented by *Barbier et al.* (Barbier et al., 2010), which focuses on how PIM models can be automatically generated from COBOL-based legacy systems.

Modernization Tools

At this time, some tools are available to deal with some ADM challenges. KDM SDK 2.0 is an Eclipse™ EMF plug-in which provides a set of tools for working with KDM. KDM SDK 2.0 is an adoption kit to help understand the KDM specification that facilitates mapping design from proprietary internal representation in the KDM metamodel and jump starts development of the KDM tools (KDMAnalytics, 2008).

MoDisco is another important tool that consists of an Eclipse™ plug-in for model-driven reverse engineering. With MoDisco, practical extractions of models can be made from various kinds of legacy systems. In addition, MoDisco proposes a generic and extensible metamodel-driven approach to model discovery (MoDisco, 2008).

Agility™, from the Obeo company, is a tool that can be used to re-document, migrate, refactor or redesign any kind of program. Agility™ supports Ada, Java, C, C++, Forte, Cobol, Oracle Forms and Visual Basic, among others type of programs (Obeo, 2007b). Furthermore, Obeo offers Acceleo™, another commercial tool that can be used together with Agility™. Acceleo™ is a code generator for transforming models into code following the MDA approach (Obeo, 2007a). Therefore, this tool makes it possible to complete a whole modernization process based on the horseshoe model.

Modernization Workbench™ makes it possible to extract technical and business knowledge from existing application portfolios. This tool creates a centralized, always-current warehouse of business and technical intelligence about the applications that run in the business (Micro Focus, 2009).

BLU AGE™ (BLU AGE, 2010) focuses on the reverse engineering and forward engineering stages. On the one hand, this tool facilitates the generation of PIM models from different platforms and languages. And on the other hand, it provides a module to generate automatically the source code of modernized systems.

Another core concept of ADM is transformation between models. There are many tools that make it possible to transform models using QVT: Medini QVT (ikv++, 2008) is an Eclipse™ application to transform models using QVT Relation, the declarative language of QVT; another tools is

SmartQVT (France Telecom R&D, 2008), which provides an Eclipse™ plug-in to perform transformations according to QVT Operational (this tool was partly financed by the ModelWare project). These tools are not alone, since many other similar tools exist. Indeed, there are other tools that enable model transformations following other transformation languages, such as ATL (ATLAS Transformation Language) (INRIA, 2005), TXL (Turing eXtender Language) (Queen's University et al., 2009), among others.

Finally, MARBLE, a non-commercial tool based on the Eclipse™ platform, can also be used to recover business processes from legacy systems in order to carry out business modernization processes (Pérez-Castillo et al., 2009a). This tool obtains business processes through three transformations: firstly, the tool recovers PSM models from different legacy software artifacts; secondly, the tool integrates those models into a single PIM model according to the KDM metamodel; and finally, MARBLE recovers a business process model by applying a set of business patterns in the KDM model (Pérez-Castillo et al., 2010).

CONCLUSION

As the history of software engineering reveals, information systems are not static entities that are unchangeable over time. Information systems degrade and age, which is known as the software erosion problem. This problem can be compared with the degradation of any material thing that deteriorates over use and over time. However, the nature of software makes it impossible to replace an information system in the same way that a material thing can be replaced. In fact, the wholesale replacement of a legacy information system has devastating economic and technical consequences.

Therefore, legacy information systems must evolve to enhanced versions of themselves, preserving the embedded business knowledge in order to extend the lifespan and improve the ROI of these legacy systems. This process is known as software modernization and the main effort to standardize this process is Architecture-Driven Modernization (ADM), as defined by the OMG.

ADM advocates carrying out reengineering processes, but treats each involved artifact as a model and establishes transformations between the models through the horseshoe model, i.e., ADM can be seen as a reengineering process that follows MDA principles. Furthermore, the ADM specification defines a wide set of standards to deal with specific modernization challenges. The standardization effort of the ADM initiative of the OMG solves the two main problems of traditional reengineering, the previous alternative to software modernization: (i) the formalization of the process in order to the modernization project can be repeatable and reusable, and (ii) the automation of the process in order to address the modernization of large complex legacy systems.

ADM can be seen as a set of standards for software modernization. The first and most important ADM standard is KDM, which offers a metamodel to represent all the software artifacts involved in a legacy system according to different views of the system and different abstraction levels. KDM facilitates the software archeology process, especially when a legacy system must be represented at a higher abstraction level, for instance in a business modernization process, since it reduces the gap between the legacy source code and high-level models like business process models.

ADM and software modernization in general have acquired a certain degree of importance within the academic and industrial communities. Indeed, there are already many ongoing modernization projects and there is a significant market for modernization tools. In addition, the main conferences on reverse engineering and software maintenance present increasingly more papers related to software modernization.

RESEARCH DIRECTION

In the future, software modernization will need *"to be addressed as a business issue as well as a technology issue, and therefore it is fundamentally interdisciplinary"* (Bennett et al., 2000). ADM will change how software and society interact, how software will be used, how software will behave, and how software will be developed. However, the biggest challenge for ADM currently is more widespread adoption in the software industry. Indeed, modernization tools cannot be effective if they are not used.

Therefore, while the launch of a modernization effort is important, the sustainability and business adoption of these efforts are equally important. Thus, the communication of the added value of a certain modernization effort is also a main challenge (Ulrich, 2010).

Moreover, in the software industry at this time, *"there is a pressing need for software on demand. This means that the basic functionality should be available before it is even required"* (Sneed, 2008). SOA environments can help to provide software as services on demand, but the problem is that a whole legacy system cannot be thrown away, and thus it must be migrated toward a SOA environment. ADM will be the cornerstone to SOA transformations since the ADM standards help in the following way (OMG, 2006b):

- They facilitate identification of redundant, inconsistent and segregated functionalities that need to be refactored to create services.
- They also identify the interfaces that could serve as prototypes for creating a service that wraps a legacy system.
- They can discover and extract business logic from legacy systems as service candidates.
- They enable necessary refactoring across multiple platforms and languages.

In another research direction, ADM will take on the most important challenges in software archeology processes, the problems of *delocalization* and *interleaving*. These problems lie in the fact that pieces of knowledge are usually scattered between many legacy systems and a single legacy system contains several pieces of knowledge (Ratiu, 2009). Therefore, in the future, KDM should support the recovery, management and deduction of different knowledge from several legacy systems in an integrated and homogeneous way.

REFERENCES

W3C. (2007). *WSDL in Web Services description working group.* Retrieved August 1, 2008, from http://www.w3.org/2002/ws/desc/

Barbier, F., Eveillard, S., Youbi, K., Guitton, O., Perrier, A., & Cariou, E. (2010). Model-driven reverse engineering of COBOL-based applications. In Ulrich, W. M., & Newcomb, P. H. (Eds.), *Information Systems transformation: Architecture-driven modernization case studies* (pp. 283–299). Burlington, MA: Morgan Kauffman.

Bennett, K. H., & Rajlich, V. T. (2000). Software maintenance and evolution: A roadmap. *Proceedings of the Conference on The Future of Software Engineering*. Limerick, Ireland: ACM.

Bianchi, A., Caivano, D., Marengo, V., & Visaggio, G. (2003). Iterative reengineering of legacy systems. *IEEE Transactions on Software Engineering*, *29*(3), 225–241. doi:10.1109/TSE.2003.1183932

BLUAGE. (2010). *BLUAGE - agile model transformation*. Netfective Technology S.A.

Canfora, G., & Penta, M. D. (2007). *New frontiers of reverse engineering. 2007 Future of Software Engineering*. IEEE Computer Society.

Chikofsky, E. J., & Cross, J. H. (1990). Reverse engineering and design recovery: A taxonomy. *IEEE Software*, *7*(1), 13–17. doi:10.1109/52.43044

de la Peyronnie, J., Newcomb, P. H., Morillo, V., Trimech, F., Nguyen, L., & Purtill, M. (2010). Modernization of the Eurocat Air Traffic Management System (EATMS). In Ulrich, W. M., & Newcomb, P. H. (Eds.), *Information Systems Transformation: Architecture driven modernization case studies* (pp. 91–131). Burlington, MA: Morgan Kauffman.

France Telecom R&D. (2008). *SmartQVT: An open source model transformation tool implementing the MOF 2.0 QVT-Operational language*. Retrieved from http://smartqvt.elibel.tm.fr/.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*. Boston: Addison Wesley Longman Publishing Co.

Ghazarian, A. (2009). A case study of source code evolution. In R. Ferenc, J. Knodel and A. Winter (Eds.), *13th European Conference on Software Maintenance and Reengineering (CSMR'09)*. (pp. 159-168). Kaiserslautern, Germany, IEEE Computer Society.

Grose, T. J., Doney, G. C., & Brodsky, S. A. (2001). *Mastering XMI: Java programming with XMI, XML, and UML*. John Wiley & Sons.

Hainaut, J.-L., Henrard, J., Hick, J.-M., Roland, D., & Englebert, V. (1996). Database design recovery. *Proceedings of the 8th International Conference on Advances Information System Engineering*, (pp. 463-480). Springer-Verlag.

Hunt, A., & Thomas, D. (2002). Software archaeology. *IEEE Software*, *19*(2), 20–22. doi:10.1109/52.991327

ikv++. (2008). *Medini QVT*. Retrieved from http://www.ikv.de/index.php?option=com_content&task=view& id=75&Itemid=77, ikv++ technologies

INRIA. (2005). *ATL transformation description template, version 0.1*. ATLAS group. Retrieved from http://www.eclipse.org/m2m/atl/doc/ATL_Transformation_Template%5Bv00.01%5D.pdf

ISO/IEC. (1992). *ISO/IEC 9075*, Database Language SQL.

ISO/IEC. (2006). *ISO/IEC 14764*, Software engineering-software life cycle processes- Maintenance. ISO/IEC. Retrieved from http://www.iso.org/iso/catalogue_detail.htm?csnumber=39064

ISO/IEC. (2009). *ISO/IEC DIS 19506*, Knowledge Discovery Meta-model (KDM), v1.1. Retrieved from http://www.iso.org/iso/catalogue_detail.htm?csnumber=32625

Jacobson, I., Christerson, M., Jonsson, P., & Övergaard, G. (1992). *Object-oriented software engineering: A use case driven approach*. Boston: Addison-Wesley.

Kazman, R., Woods, S. G., & Carrière, S. J. (1998). Requirements for integrating software architecture and reengineering models: CORUM 2. *Proceedings of the Working Conference on Reverse Engineering (WCRE'98)*, (pp. 154-163). IEEE Computer Society.

KDMAnalytics. (2008). *Knowledge Discovery Metamodel (KDM) software development kit 2.0 eclipse plugin*. Retrieved from http://www.kdmanalytics.com/kdmsdk/KDMSDK_brochure.pdf, Hatha Systems.

Khusidman, V. (2008). *ADM transformation White Paper. DRAFT V.1*. Retrieved from http://www.omg.org/docs/admtf/08-06-10.pdf

Khusidman, V., & Ulrich, W. (2007). *Architecture-driven modernization: Transforming the enterprise. DRAFT V.5*. Retrieved from http://www.omg.org/docs/admtf/07-12-01.pdf

Koskinen, J., Ahonen, J., Lintinen, H., Sivula, H., & Tilus, T. (2004). *Estimation of the business value of software modernizations*. Information Technology Research Institute, University of Jyväskylä.

Koskinen, J., Ahonen, J. J., Sivula, H., Tilus, T., Lintinen, H., & Kankaanpää, I. (2005). Software modernization decision criteria: An empirical study. *European Conference on Software Maintenance and Reengineering,* (pp. 324-331). IEEE Computer Society.

Lehman, M. M., Perry, D. E., & Ramil, J. F. (1998). Implications of evolution metrics on software maintenance. *Proceedings of the International Conference on Software Maintenance*, (pp. 208-217). IEEE Computer Society.

Melton, J., & Simon, A. R. (1993). *Understanding the new SQL: A complete guide*. USA: Morgan Kaufmann Publishers, Inc.

Mens, T., & Demeyer, S. (2008). *Software Evolution*. Berlin, Heidelberg: Springer-Verlag.

Micro Focus. (2009). *Modernization Workbench*™. Retrieved from http://www.microfocus.com/products/modernizationworkbench/

Miller, J., & Mukerji, J. (2003). *MDA Guide Version 1.0.1.* Retrieved from www.omg.org/docs/omg/03-06-01.pdf

ModelWare. (2006). *MODELWARE.* Retrieved August 24, 2009, from http://www.modelware-ist.org/

MoDisco. (2008). *KDM-to-UML2 converter*. Retrieved from http://www.eclipse.org/gmt/modisco/toolBox/KDMtoUML2Converter/

MOMOCS. (2008). *MOdel driven MOdernisation of Complex Systems is an EU-Project.* Retrieved from http://www.momocs.org/. 2008, from http://www.momocs.org/

Moyer, B. (2009). Software archeology: Modernizing old systems. *Embedded Technology Journal*, *1*, 1–4.

Müller, H. A., Jahnke, J. H., Smith, D. B., Storey, M.-A., Tilley, S. R., & Wong, K. (2000). Reverse engineering: A roadmap. *Proceedings of the Conference on The Future of Software Engineering*. Limerick, Ireland: ACM.

Obeo. (2007a). *Acceleo*™. Retrieved from http://www.obeo.fr/pages/acceleo/en

Obeo. (2007b). *Agility*™. Retrieved from http://www.obeo.fr/pages/agility/en

OMG. (2003a). *CWM. Common Warehouse Metamodel, v1.1 Specification.* Retrieved from http://www.omg.org/spec/CWM/1.1/PDF/

OMG. (2003b). *Why do we need standards for the modernization of existing systems?* OMG ADM task force.

OMG. (2006a). *ADM Glossary of Definitions and Terms.* Retrieved from http://adm.omg.org/ADM_Glossary_Spreadsheet_pdf.pdf

OMG. (2006b). *Architecture-driven modernization scenarios.* Retrieved November, 2, 2009, from http://adm.omg.org/ADMTF_Scenario_White_Paper%28pdf%29.pdf

OMG. (2007a). *ADM task force by OMG.* Retrieved June 15, 2009, 2008, from http://www.omg.org/

OMG. (2007b). *Unified modeling language: Superstructure, version 2.0.* Retrieved August 16, 2007, from http://www.omg.org/docs/formal/05-07-04.pdf

OMG. (2008a). *Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM), v1.0.* Retrieved from http://www.omg.org/docs/formal/08-01-01.pdf

OMG. (2008b). *QVT. Meta Object Facility (MOF) 2.0 query/vew/transformation specification.* Retrieved from http://www.omg.org/spec/QVT/1.0/PDF

OMG. (2009a). *Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM), v1.1.* Retrieved from http://www.omg.org/spec/KDM/1.1/PDF/

OMG. (2009b). *Architecture-Driven Modernization standards roadmap.* Retrieved October 29, 2009, from http://adm.omg.org/ADMTF%20Roadmap.pdf

OMG. (2009c). *UML (Unified Modeling Language) superstructure specification. Version 2.2.* Retrieved June 24, 2009, from http://www.omg.org/spec/UML/2.2/Superstructure/PDF/

Paradauskas, B., & Laurikaitis, A. (2006). Business knowledge extraction from legacy Information Systems. *Journal of Information Technology and Control*, *35*(3), 214–221.

Pérez-Castillo, R., García-Rodríguez de Guzmán, I., Ávila-García, O., & Piattini, M. (2009a). MARBLE: A modernization approach for recovering business processes from legacy systems. *International Workshop on Reverse Engineering Models from Software Artifacts (REM'09)*, Lille, France.

Pérez-Castillo, R., García-Rodríguez de Guzmán, I., Ávila-García, O., & Piattini, M. (2010). Business process patterns for software archeology. *25th Annual ACM Symposium on Applied Computing (SAC'10)*. Sierre, Switzerland: ACM.

Pérez-Castillo, R., García Rodríguez de Guzmán, I., Caballero, I., Polo, M., & Piattini, M. (2009b). PRECISO: A reengineering process and a tool for database modernisation through Web services. *24th Annual ACM Symposium on Applied Computing. Track on Service Oriented Architectures and Programming (SOAP 2009),* (pp. 2126-2133). Waikiki Beach, Honolulu, Hawaii: ACM.

Polo, M., Piattini, M., & Ruiz, F. (2003). *Advances in software maintenance management: Technologies and solutions*. Hershey, PA: Idea Group Publishing.

Queen's University. NSERC & IBM. (2009). Turing eXtender Language (TXL). Retrieved from http://www.txl.ca/

Ratiu, D. (2009). Reverse engineering domain models from source code. *International Workshop on Reverse Engineering Models from Software Artifacts (REM'09)*. Lille, France.

Sneed, H. M. (2005). *Estimating the Costs of a Reengineering Project*. IEEE Computer Society.

Sneed, H. M. (2008). *Migrating to Web services. Emerging Methods, Technologies and Process Management in Software Engineering* (pp. 151–176). Wiley-IEEE Computer Society Pr.

Sommerville, I. (2006). *Software Engineering*. Addison Wesley.

Ulrich, W. M. (2002). *Legacy systems: Transformation strategies*. Prentice Hall.

Ulrich, W. M. (2010). Launching and sustaining modernization initiatives. In Ulrich, W. M., & Newcomb, P. H. (Eds.), *Information Systems Transformation: Architecture driven modernization case studies* (pp. 403–418). Burlington, MA: Morgan Kauffman.

Visaggio, G. (2001). Ageing of a data-intensive legacy system: Symptoms and remedies. *Journal of Software Maintenance*, *13*(5), 281–308. doi:10.1002/smr.234

ADDITIONAL READING

Newcomb, P. (2005). Architecture-Driven Modernization (ADM). Proceedings of the 12th Working Conference on Reverse Engineering, IEEE Computer Society. Heuvel, W.-J. v. d. (2006). Aligning Modern Business Processes and Legacy Systems: A Component-Based Perspective (Cooperative Information Systems), The MIT Press.

Ulrich, W., & Newcomb, P. (2010). *Information Systems Transformation. Architecture Driven Modernization Case Studies*. Burlington, MA: Morgan Kauffman.

2.2. Knowledge Discovery Metamodel

This section focuses on the use of the KDM standard to modernize legacy information systems. This section focuses on the use of KDM as a common repository to represent all the business knowledge embedded in legacy information systems. This section also introduces how KDM models can be used to retrieve business processes from information systems.

The content of this section is a paper published in the Computer Standards and Interfaces Journal. This journal had an impact factor of 0.825 in 2010, according to Thomson's Science Citation Index (SCI). The reference of this paper is: *Pérez-Castillo, R., I.G.-R. deGuzmán, and M. Piattini, Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems. Computer Standards & Interfaces, 2011. 33(6): p. 519-532.*

Computer Standards & Interfaces 33 (2011) 519–532

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems

Ricardo Pérez-Castillo *, Ignacio García-Rodríguez de Guzmán, Mario Piattini

Alarcos Research Group, University of Castilla-La Mancha, P° de la Universidad, 4 13071, Ciudad Real, Spain

ARTICLE INFO

Article history:
Received 27 April 2010
Received in revised form 15 February 2011
Accepted 21 February 2011
Available online 27 February 2011

Keywords:
KDM
ISO19506
Knowledge management
Software modernization
Legacy system

ABSTRACT

Legacy systems age over time as a consequence of uncontrolled maintenance, thus they must be evolved while its valuable embedded knowledge is preserved. Software modernization, and particularly Architecture-Driven Modernization, has become the best solution in the legacy systems' evolution. ADM defines the Knowledge Discovery Metamodel specification, now being adopted as ISO/IEC 19506 by the International Standards Organization. The KDM metamodel allows to represent all the software artifacts recovered during reverse engineering techniques at different abstraction levels. This paper presents how to use KDM to modernize legacy systems, making them more agile, preserving the embedded business knowledge and reducing maintenance costs.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

As the history of software engineering reveals, information systems are not static entities, but changeable over time. Information systems degrade and age, and they become *legacy* information systems because the code of these systems was written long ago and now may be technologically obsolete [49]. This problem is known as the software erosion phenomenon [52]. Most often, it is due to maintenance activities carried out over time, since the successive maintenance changes in an information system minimize its quality. In this case, a new and improved system must replace the previous one. However, replacing these systems completely from scratch is very expensive, and also slows down the achievement of ROI (Return of Investment) [48]. Additionally, the software embeds a significant amount of business knowledge over time that would be lost if entirely replaced [44].

In tackling the software erosion phenomenon, evolutionary maintenance is the best solution for obtaining improved systems without discarding the existing systems, which minimizes software erosion effects. Evolutionary maintenance addresses adaptive and perfective maintenance changes [17], and makes it possible to manage controllable costs and preserve the valuable business knowledge embedded in the legacy system.

Reengineering has been the main mechanism for addressing the evolutionary maintenance of legacy systems for a considerable time [8]. Reengineering preserves the legacy knowledge of the systems and makes it possible to maintain software easily at a low cost [7]. This reengineering process consists of the examination and alteration of a legacy system to reconstitute it in a new form and the subsequent implementation of the new form [10].

Nevertheless, a 2005 study [48] states that over 50% of all reengineering projects currently fail, due to two main weaknesses: formalization and automation problems. On the one hand, reengineering processes lack formalization and standardization [21], since these processes are carried out in an *ad hoc* manner. Thus, different reengineering tools that address specific tasks in the reengineering process cannot be integrated or reused in different reengineering projects. And on the other hand, the reengineering of large complex legacy information systems is very difficult to automate [9]; since the reengineering processes cannot be repeatable, as a consequence of the formalization problem. Therefore the cost of maintenance based on reengineering grows significantly.

At this time, software modernization is a new specific kind of evolutionary maintenance paradigm to solve reengineering problems. ADM (Architecture-Driven Modernization) as defined by OMG (Object Management Group) [36], advocates carrying out the reengineering process but considering model-driven development principles. That is, ADM deals with all the involved software artifacts as models in a homogenous manner, and it facilitates the formalization of deterministic transformations between those models [33]. The model transformations can be formalized by means of QVT (Query/Views/Transformations), the model transformation language proposed by OMG [39]. The automation of the model transformations together with the model-driven development principles makes it

* Corresponding author. Tel.: +34 926295300; fax: +34 926295354.
E-mail addresses: ricardo.pdelcastillo@uclm.es (R. Pérez-Castillo), ignacio.grodriguez@uclm.es (I.G.-R. de Guzmán), mario.piattini@uclm.es (M. Piattini).

0920-5489/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2011.02.007

possible to reuse the models involved in ADM-based processes. As a consequence, the automation problem can also be solved due to the automated transformations together with the reuse of the models.

As part of the effort undertaken by the ADM Task Force of OMG, KDM (Knowledge Discovery Metamodel) is the first standard within a broad set of proposed standards [43], although many of these standards are still in the approval or development stage. KDM forms the core of that set of standards for two main reasons: (i) KDM addresses the main challenges that appear in the modernization of legacy information systems; and (ii) it is the cornerstone of the set of proposed standards, since the other standards are defined around KDM. Despite the fact that KDM had been defined by the OMG, it is being adopted as the international standard ISO/IEC 19506 ADM KDM [19] through the so-called "fast-track" liaison relationship between OMG and ISO.

KDM is a specification of agreed upon facts and how these facts are represented in XML by using OMG standards such as Model Object Facility (MOF) and XML Metadata Interchange (XMI). The availability of this information represented according to the KDM specification makes it possible (i) to store the facts about information systems in a compliant structure; (ii) to analyze and reason about KDM facts; (iii) to exchange KDM facts as XML documents; and (iv) to build tools so that parsing source code in a given programming language to generate language-independent and vendor-neutral facts about information systems.

From the modernization perspective, KDM allows addressing all the stages of modernization processes based on ADM: reverse engineering, restructuring and forward engineering [24]. In the reverse engineering stage, the software modernization process analyzes the legacy system (at a lower abstraction level) in order to identify the components of the system and their interrelationships and build one or more representations of the legacy system (at a higher abstraction level). In addition, the metamodel defined by the KDM standard provides a common repository structure that makes it possible to exchange information in the restructuring and forward engineering stages about all existing software artifacts in legacy systems like source codes, databases, user interfaces, business rules, etc. [42].

The representation of the system's knowledge throughout the software modernization stages has four key challenges that must be addressed:

- Legacy systems must be represented from different viewpoints and different abstraction levels where it must be possible to represent all the software artifacts in legacy systems [20]. In addition, the different views of the systems must be linked in order to support the traceability between elements at different views or levels.
- The knowledge embedded in legacy systems must not only be represented in a suitable and accurate way in the reverse engineering stage, but must also be managed throughout the following stages in the whole software modernization process: the restructuring and forward engineering stages.
- It must be possible for different tools to share the recovered knowledge in order to automate the different modernization tasks.
- It must be possible to discover or deduce new implicit knowledge from the explicit knowledge recovered from legacy systems.

All these challenges are met by KDM standard ISO/IEC 19506. This paper presents in detail how the KDM standard can address the challenges presented. Moreover, this paper shows the way in which KDM assists in the entire software modernization process in order to minimize the effects of software erosion on legacy systems.

The remainder of this paper is structured as follows: Section 2 presents a brief history of KDM in order to understand the role that KDM plays in the modernization of legacy systems. Section 3 summarizes works related to the knowledge representation used in software modernization and similar processes. Section 4 presents the elements and features of the metamodel defined by the KDM standard. In addition, this section presents the KDM standard from an ontological perspective and as a common interchange format. Section 5 shows how KDM can be used in ADM-based processes in order to exploit all their benefits. Finally, Section 6 summarizes and discusses the conclusions of this paper.

2. History of KDM

In June 2003, OMG formed a task force to model software artifacts in the context of legacy systems. Initially, the group was called the *Legacy Transformation Task Force*, but soon, the group changed its name to the *Architecture-Driven Modernization Task Force* (ADMTF). In July 2003, the ADMTF issued a software modernization whitepaper [34].

In November 2003, the ADMTF issued the request-for-proposal of the Knowledge Discovery Metamodel (KDM) specification. The objective of this initial metamodel was to provide a comprehensive view of the application structure and data, but it did not represent software below the procedure level. The request-for-proposal stated that the metamodel of the KDM standard:

- represents artifacts of legacy software as entities, relationships and attributes
- includes external artifacts with which software artifacts interact
- supports a variety of platforms and languages
- consists of a platform and language independent core, with extensions where needed
- defines a unified terminology for legacy software artifacts
- describes the physical and logical structure of legacy systems
- can aggregate or modify, i.e. refactor, the physical system structure
- facilitates tracing artifacts from logical structure back to physical structure
- represents behavioral artifacts down to, but not below, the procedural level.

In May 2004, six organizations responded to the request-for-proposal. However, throughout 2004 and 2005 more than 30 organizations from five different countries have collaborated in the development and review of the KDM standard. In May 2006, KDM was adopted by OMG and moved into the finalization stage in the adoption process. In March 2007, OMG presented the recommended specification of KDM 1.0. In April 2007, OMG started ongoing maintenance of the KDM specification.

In January 2009, the recommended specification of KDM 1.1 was published by OMG [42], and in addition, OMG started the revision of that version. Recently, in March 2009, ISO started the adoption process of the KDM specification, which is known as ISO/IEC ADM KDM 19506 [19].

3. Related work

Knowledge management based on models throughout all the stages of software modernization has been widely studied. Several alternative solutions were proposed prior to the KDM standard.

In most cases, the knowledge recovered through reverse engineering techniques is represented and managed in an *ad hoc* way. Zou et al. [55] propose a framework based on a set of heuristic rules for extracting business processes through the static analysis of the source code. The authors of this work also use an algebraic approach in the study of business processes. Favre [13] proposes an MDA-based framework for modernizing legacy systems. This work defines NERUS, an *ad hoc* metamodeling language, to represent metamodels and metamodel transformations. Perez-Castillo et al. [47] propose an ADM-based framework to generate web services from relational database schemas. The authors of this work use a specific metamodel according to the SQL standard to represent the knowledge about

database schemas. Contrary to the previous framework, Vara et al. [51] propose a framework to obtain object relational database schemas from UML models. Maciaszek [25] proposes the meta-architecture called PCMEF (Presentation, Control, Mediator, Entity, and Foundation) to support roundtrip architectural modeling as an alternative solution to MDA. Grau et al. [15] propose PRiM, an i^*-based business process reengineering method for information systems specification. This *ad hoc* method uses the i^* framework [54] which proposes the use of two types of models: (i) the SD (Strategic Dependency) model which represents the intentional concepts of a process; and (ii) the SR (Strategic Rationale) model which represents the rationale behind it. Daga et al. [11] present an ontological approach for recovering legacy business knowledge for modernizing legacy systems. The proposed ontology not only takes the technical aspect of the legacy systems into account, but also considers the business knowledge which resides within the system. Finally, another relevant work is the knowledge management framework proposed by Yang et al. [53]. This framework carries out a hybrid peer-to-peer knowledge management architecture to consider the tradeoff between centralized and decentralized knowledge in a business environment. The approach is based on the transactive memory theory, i.e., the collective memory of small business groups.

Several works opt for the *ad hoc* option because this kind of solution is more suitable to the specific needs in each case. However, the most important problem with *ad hoc* solutions is the standardization problem, since these frameworks or approaches cannot be used in the same way for all kinds of systems, all kinds of domains, all kinds of technologies, etc.

Due to the formalization problem, other authors have used standards to represent and manage the knowledge in the reverse engineering stage. A common standard used in some works is UML. Armour et al. [4] show how to use standard UML models to capture architectural view knowledge from legacy systems. In addition, the authors propose specific UML extensions in order to address the representation of business models. Abdullah et al. [1] propose an extension to UML for knowledge modeling based on the profiling extension mechanism of UML. This UML profile focuses on the knowledge management in knowledge-based systems.

Despite these works, UML was not intended to be used as a metamodel to represent all the artifacts involved in a legacy system that is being modernized. Indeed, UML has several limitations since it does not support the representation of the business view of legacy systems at higher abstraction level. Particularly, another standard called SBVR [40] (Semantics of Business Vocabulary and Business Rules) makes it possible to represent and manage business knowledge. Some works try to recover business logic from legacy systems using this standard. Nelson et al. [32] present a lifecycle of the management of business rules based on the SBVR standard and they compare it to the traditional knowledge management lifecycle. Moreover, Muehlen et al. [31] compare the SBVR with other standards with similar purposes such as SRML (Simple Rule Markup Language) and BPMN (Business Process Model and Notation), and analyze the limitations of the SBVR standard. The main limitation of SBVR is that it does not support the representation of events.

All these proposals are limited to managing the knowledge involved in the reverse engineering stage. On the one hand, some proposals are not adequate due to the fact that they are not standards, neither *de facto* nor *de jure*. And on the other hand, the standard-based proposals focus on the technical view or business view of the legacy system in an isolated way. The KDM standard solves this problem because it is standard, and in addition, KDM makes it possible to represent software artifacts of a technical and business nature at different abstraction levels. For this reason, KDM is the most suitable solution to represent and manage the knowledge involved in any modernization process to minimize the effects of software erosion. The next section presents the KDM standard in detail.

4. Knowledge Discovery Metamodel

The goal of the KDM standard is to define a metamodel to represent all the different legacy software artifacts involved in a legacy information system (e.g. source code, user interfaces, databases, etc.). The metamodel of the KDM standard provides a comprehensive high-level view of the behavior, structure and data of legacy information systems by means of a set of facts. The main purpose of the KDM specification is not the representation of models related strictly to the source code nature such as UML (Unified Modeling Language) [19]. While UML can be used to generate new code in a *top-down* manner, an ADM-based process using KDM starts from the different legacy software artifacts and builds higher-abstraction level models in a *bottom-up* manner through reverse engineering techniques [30].

The KDM specification can be seen from different perspectives. Firstly, KDM can be considered as a metamodel to represent legacy knowledge models. Secondly, most of the KDM specification is a definition of a language- and platform-independent ontology of legacy information systems. Finally, KDM can be also seen as a common interchange format that makes the interoperability between the reverse engineering tools and modernization tools possible. The next subsections show KDM from each perspective.

4.1. KDM as a metamodel

KDM defines an MOF-compliant metamodel, which means that the metamodel of the KDM standard is defined as an Entity-Relationship model according to the MOF (Meta Object Facility) meta-metamodel [37]. Several metamodels standardized by OMG, including UML, CWM (Common Warehouse Metamodel) and SPEM (Software & System Process Engineering Metamodel) [41] use MOF as KDM does.

In order to manage the complexity of KDM, the structure of the metamodel of the KDM standard is divided into several layers representing both physical and logical software artifacts of legacy information systems at different abstraction levels. Furthermore, each abstraction layer separates the knowledge about legacy information systems into various orthogonal concerns that are well-known in software engineering as architecture views [19]. Fig. 1 shows the organization of the metamodel of the KDM standard.

The metamodel of the KDM standard defines four layers: *infrastructure*, *program elements*, *runtime resources* and *abstractions* layer. Each abstraction layer of the metamodel is based on the previous layer. Furthermore, each layer is organized into packages that define a set of metamodel elements whose purpose is to represent a specific independent concern of knowledge related to legacy systems.

4.1.1. Infrastructure layer

The *infrastructure* layer, at the lowest abstraction level, defines a small set of concepts used systematically throughout the entire KDM specification. There are three packages in this layer: *core*, *KDM* and *source*.

4.1.1.1. Core package. The *core* package defines the basic constructs for creating and depicting the metamodel elements in all KDM packages. This means that the *core* package determines the structure of the KDM models and defines the fundamental patterns and constraints implemented by all the other KDM packages.

Therefore, any metamodel element defined in any other package must be a subclass of one of the *core* classes. The two fundamental classes (metamodel elements) of the *core* package are *KDMEntity* and *KDMRelationship*, since KDM is an entity-relationship representation. A KDM entity "is an abstraction of some element of an existing software system that has a distinct, separate existence, a self-contained piece of data that can be referenced as a unit" [19]. Moreover, a KDM relationship "represents some semantic association between elements of an existing software system" [19].

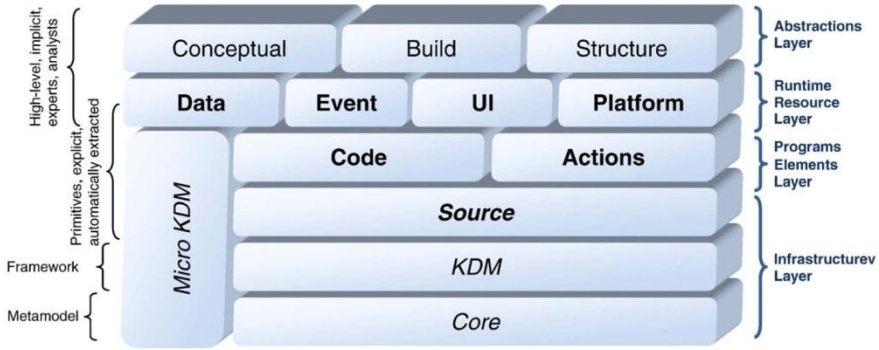


Fig. 1. Layers, packages, and concerns in KDM (Adapted from [19]).

In addition, the *core* package defines the data types that are used to represent all the different models according to packages defined by the KDM specification. Only three types exist: string, integer and boolean, which are subclasses of the MOF *DataType* class. These are not related to the wide variety of datatypes in the system of interest that is being represented by KDM. These others data types are represented by means of particular elements of the *Code* package (cf. Section 4.1.2).

4.1.1.2. KDM package. The *KDM* package also defines the common metamodel elements that constitute the infrastructure for other packages of the metamodel. In addition, the KDM package defines the elements that constitute the framework of each KDM representation. This framework determines the physical structure of a KDM representation (also referred to as KDM instances), i.e., each KDM representation consists of one or more *Segment* elements that own several KDM models.

Each *KDM* package defines some specific kind of KDM model, which addresses a specific concern in the knowledge about the legacy systems. Thus, each specific implementation of KDM might support one or more selected KDM models grouped into *Segment* elements. The KDM implementers are responsible for defining the KDM models used in each case to represent the elements of programming languages, runtime platforms, and any other legacy software artifact.

In addition, the *KDM* package defines the common lightweight extension mechanism to add new metamodel elements in order to extend the semantic in specific KDM representations. This mechanism consists of introducing new types of extended metamodel elements, also known as stereotypes. The stereotypes modify the semantic of a certain element.

The *KDM* package, together with the *core* package, constitutes the so-called KDM framework. These packages do not define any kind of KDM model since they represent the common infrastructure for the rest of the KDM specification.

4.1.1.3. Source package. The *source* package defines the first and most basic model of KDM: the *inventory* model. This model enumerates the physical artifacts of a legacy system (like source files, images, configuration files, resource files, and so on) as KDM entities and defines a traceability mechanism to link those KDM entities and their original representation in the legacy source code, for which the KDM representation was created. Other KDM models in other KDM packages use this mechanism to refer to the physical software artifacts.

Fig. 2 shows the metamodel elements of the inventory model in the *source* package. The *InventoryModel* element represents the root metamodel element in the *source* package. The *InventoryModel* has several *AbstractInventoryElements*, and in turn, it has several *AbstractInventoryRelationships* (both elements are, respectively, subclasses of *KDMEntity* and *KDMRelationship*).

The *AbstractInventoryElement* is specialized into concrete metamodel elements in order to represent different physical artifacts: *BinaryFile*, *ExecutableFile*, *ResourceDescription*, *Configuration* and *Image*. Also, *AbstractInventoryElement* is specialized in container elements such as *Directory* and *Project* that group other inventory elements.

The *source* package offers two mechanisms for linking a KDM element to the corresponding physical artifact, on the one hand, the sequence enumeration of the corresponding fragments of source code (*Segment* element of the *KDM* package) into a KDM representation, and on the other hand, the linkage of a KDM element to a specific region of the source code (*SourceRegion* element) within some physical artifact (*SourceFile* element) by means of a reference element (*SourceRef* element).

Moreover, the inventory elements are interrelated by means of *AbstractInventoryRelationships* (see Fig. 2), which is specialized into *DependsOn* and *InventoryRelationship* elements. *InventoryRelationship* is a generic relationship between any inventory element and any KDM entity in any kind of KDM model. However, the *DependsOn* element represents a relationship between two inventory elements. The *DependsOn* relationship means that an inventory model requires another inventory element during one or more steps of the engineering process. For instance, imagine an executable file that needs a certain configuration file.

4.1.2. Program elements layer

The *program elements* layer offers a broad set of metamodel elements in order to provide a language-independent intermediate representation for various constructs defined by common programming languages. This layer represents implementation level program elements and their associations. This means that the *program elements* layer represents the logical view of a legacy system. This layer has two packages: the *code* package and the *action* package. Both packages define a single model, called the *CodeModel*.

4.1.2.1. Code package. The *code* package defines a set of *CodeItem* elements that represents the common named elements in the source code supported by different programming languages such as data types, classes, procedures, methods, templates and interfaces. Also, the *code* package makes it possible to represent the structural relationships between the code elements.

Fig. 3 shows the metamodel elements of the code model. The *CodeModel* element aggregates several *CodeItem* elements. The *CodeItem* element is specialized into three major metamodel elements: *Module*, *ComputationalObject* and *DataType*.

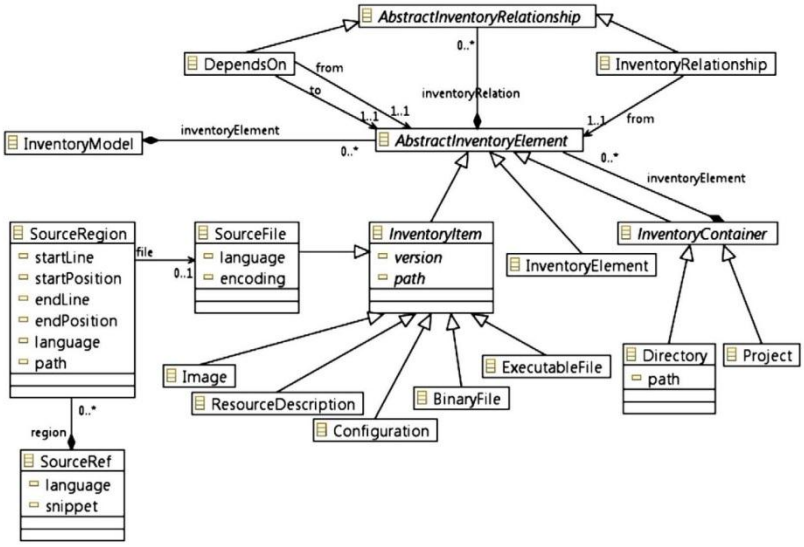


Fig. 2. The inventory model in the KDM source package (adapted from [19]).

- The *Module* element represents "a discrete and identifiable program unit that contains other program elements and may be used as a logical component of the software system" [19]. The *Module* element is specialized into *Package*, *CompilationUnit*, *CodeAssembly*, etc.
- The *ComputationalObject* element and its subclasses represent callable computational objects like *MethodUnits* or *CallableUnits* in general.
- The *DataType* metamodel element is a common superclass that defines the named data items of legacy systems such as global/local variables, record files and formal parameters in callable units. In addition, the data representation in KDM is aligned with the General-Purpose Datatypes (GPD) standard [18]. This standard defines the nomenclature and shared semantics for various datatypes commonly used in programming languages.

Moreover, the code elements that make up the code model can have several code relationships. These relationships are defined by means of the *AbstractCodeRelationship* element and its subclasses: *Imports*, *Extends*, *Implements*, *Include*, among others.

In conclusion, the *code* package provides broad coverage for most common code elements and data types of several programming languages. Nevertheless, the metamodel of the KDM standard also offers the generic extensible mechanism based on stereotypes to

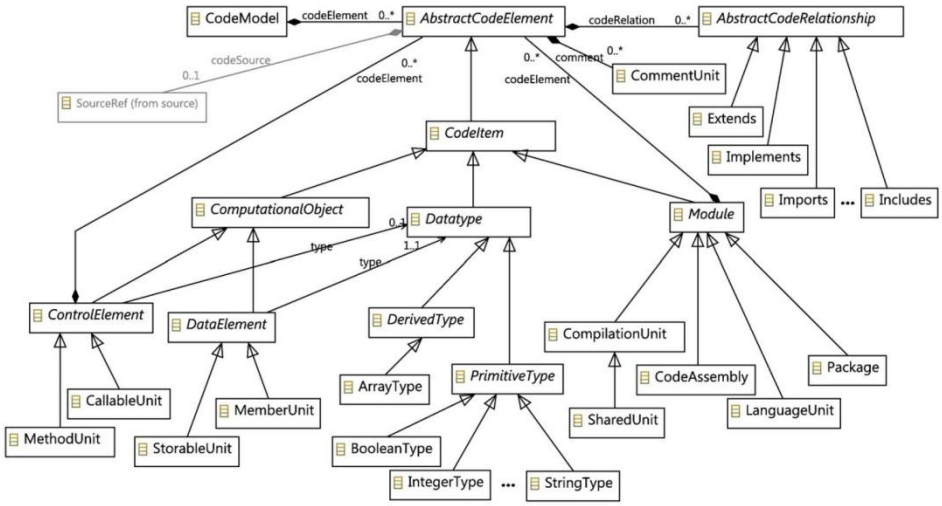


Fig. 3. The code model in the KDM code package (adapted from [19]).

represent those uncommon code fragments not supported by the *code* package.

4.1.2.2. Action package. The *action* package extends the code model by means of more metamodel elements that represent behavior descriptions and control-and-data-flow relationships between code elements. The *action* package adds two key elements: the *ActionElement* and the *AbstractActionRelationship* (see Fig. 4).

The *ActionElement* metamodel element depicts a basic unit of behavior and represents some programming language constructs such as statements, operators and conditions. The feature *kind* in this element specifies the precise semantic of the action according to a finite set of action types defined in the *Micro KDM*, which categorizes the possible actions in legacy information systems by providing additional semantic to each action. Also, the *ActionElements* can be linked to the original representation in legacy code by means of the *SourceRef* element of the *source* package.

The *AbstractActionRelationship* usually represents the use of a name in a statement. This use is done in two ways: this kind of element can represent (i) a control flow between two actions (*ControlFlow* element and its subclasses) or (ii) an association from an action element to a code element in the code model. For example, the specialized element *Calls* refers an *ActionElement* (feature *from*) and also refers a *ControlElement* that typically can be a *CallableUnit* (feature *to*). The *Call* element represents an invocation from a sentence in the legacy code to a callable unit (a method or function). Other specialized elements that are similar to the *Call* element are the *Writes* and *Reads* elements in order to represent the access to data and the *ExceptionFlow* element which represents the anomalous control flows in the source code, among other metamodel elements (see Fig. 4).

4.1.3. Runtime resource layer

The *runtime resource* layer enables the representation of high-value knowledge about legacy systems and their operating environment, i.e., this layer focuses on those things that are not contained within the code itself.

The *runtime resource* layer focuses on representing resources managed by the runtime platform. For this purpose, this layer provides abstract resource actions to manage the resources. Each package in this layer defines specific entities and containers to represent all the resources of the specific concerns of legacy systems. Also, the packages in this layer define specific structural relationships between the resources as well as specific resource actions to depict manipulations of resources related to the metamodel elements of previous KDM layers.

The *runtime resource* layer establishes four packages: *data*, *event*, *UI* and *platform*. In turn, these four packages respectively define four KDM models with the same name.

4.1.3.1. Data package. The *data* package defines the representation of several data organization capabilities. However, this representation is related to the persistent data aspects of a legacy system, since the application data as input/output variables or parameters in callable units are represented through the *code* package. Therefore, this package makes it possible to represent complex data repositories like relational databases, record files and XML schemas and documents.

The *data* package defines *DataModel* in order to represent all the persistent data resources of a specific legacy system (*DataResource* metamodel elements) as well as the actions with those elements (*DataAction* metamodel elements).

The *DataResource* element is specialized into several metamodel elements. Regarding the relational database representation, *DataResource* is specialized into *Catalog* and *RelationSchema* elements. In turn, the *RelationSchema* element can aggregate several *ColumSet* elements, which are specialized into *DataSegment*, *RelationalTable*, *RelationalView*, *RecordFile*, and so on. In addition, the *DataResource* element is specialized into the *IndexElement*, which is also specialized into the *Index*, *UniqueKey* or *ReferenceKey* element.

Also, the *data* package defines a set of data elements regarding the XML schema representation. *AbstractContentElement* represents any XML schema definition and is specialized into *ContentItem*, *ComplexContentItem* and *ContentRestriction*. The *ContentItem* element is specialized into *ContentElement*, *ContentAtribute*, *ContentReference*, etc. *ComplexContentItem* is also specialized to represent *SimpleContent*, *MixedContent*, *SeqContent*, *ChoiceContent*, and so on. Finally, *ContentRestriction* defines XML schema restrictions like *minExclusive*, *length*, and *whitespace*, among others.

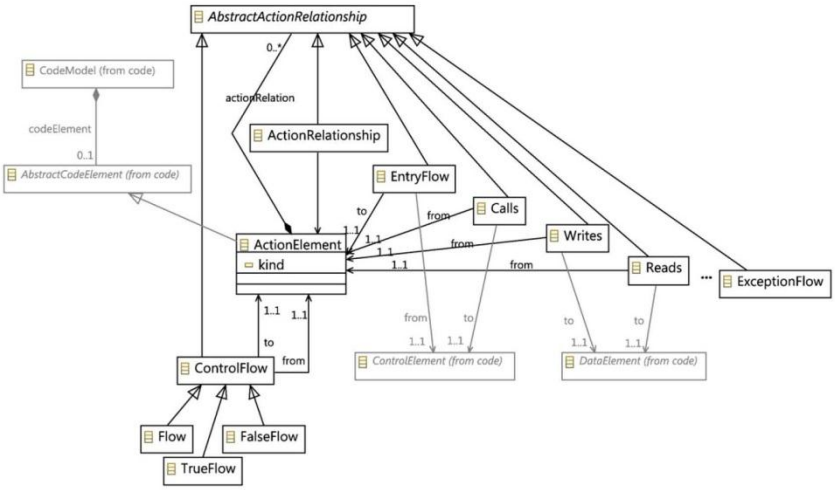


Fig. 4. The code model in the KDM action package (adapted from [19]).

Moreover, the *data* package defines data actions since it extends the *AbstractActionRelationship* element of the *action* package with four new elements. These new action relationships are established from an *ActionElement* in the code model to a *DataResource* element in the data model. The four new relationships are the following: (i) the *ReadColumnSet* element represents a flow of data from the data resource, for instance this element makes it possible to model the *select* SQL queries embedded in the source code; (ii) the *WritesColumnSet* element represents a flow of data to the data resource, for instance it models the *insert* queries; (iii) the *ManageData* element represents the data accesses when there is no flow of data to or from the data resources, thus the nature of the operation is represented by the *kind* attribute related to the *micro KDM* package; and finally (iv) the *HasContent* element represents a structural relationship that defines a linkage between the action element and the data resource.

4.1.3.2. Event package. The *event* package defines a set of metamodel elements for representing state and state transitions caused by events. This package represents two kinds of states: (i) the concrete states that are explicitly supported by specific state-machine based languages, such as CHILL [16]; and (ii) abstract states related to a specific algorithm, resource or user interface.

The *event* package defines the *EventModel* metamodel element that aggregates the set of *EventResources* of a legacy system. The *EventResource* metamodel element is specialized into *State*, *Transition* and *Event* elements. In addition, each *EventResource* element is related to one or more *AbstractEventRelationships* and this element is specialized into two relationships: the *NextState* metamodel element establishes a relationship from a *Transition* to the given *State* after this event-based transition, and *ConsumeEvent* defines a relationship between a *Transition* and the *Event* that triggers that transition.

Moreover, the *event* package extends the *action* package since it defines a set of event actions that can be used with the *EventResources*. These event actions include the *ReadState* element which represents the access to the current state from a specific code element. The *ProducesEvent* element is similar to the *Writes* relationship of the *action* package, since it defines the event that is produced in a concrete code element. Finally the *HasState* element represents a structural relationship and makes it possible to associate an element of an event model with any resource.

4.1.3.3. UI package. The *UI* package provides metamodel elements for representing the resources related to user interface aspects, such as their composition, their sequence of operations and their relationships to the legacy systems.

The *UI* package also defines a specific model through the *UIModel* element in order to statically represent the main components of the user interface of a legacy system. The UI model consists of a set of *UIResource* metamodel elements, and in turn, the *UIResource* element is specialized into *UIDisplay*, *UIField*, *UIEvent*, among others. The UI relationships defined by this package are only two: *UILayout* represents the behavior of the user interface as the sequential flow from one instance of display to another; and *UILayout* establishes a relationship between a portion of a user interface and its layout.

This package provides a set of actions that extends the action elements defined by the *action* package. These action metamodel elements are: *ManageUI*, *ReadsUI* and *WritesUI* and their semantic is very similar to the semantic defined by the *action* package, but these action elements refer to *UIResource* elements.

4.1.3.4. Platform package. The *platform* package defines artifacts related to the runtime platform and environment of a legacy system such as inter-process communication, the use of registries, the management of data, and so on. The platform metamodel elements depict the execution context of the legacy code, since a legacy system is not only determined by the programming language of the source code, but also by the selected runtime platform.

The *platform* package defines the *PlatformModel* element as an aggregation of *AbstractPlatformElements* that is specialized into specific metamodel elements like *Process*, *Thread*, *DataManager*, *FileResource*, *StreamResource*, *PlatformEvent*, *NamingResource*, etc. In addition, this model defines only one platform relationship named *BindTo* that defines an association between two platform resources.

Regarding the actions, the platform model extends the *action* package with four action metamodel elements: *DefinedBy*, *ManagesResource*, *WritesResources* and *ReadsResources*. The semantic of these action elements is very similar to the elements defined by the *action* package, although the actions refer to the platform resources.

4.1.4. Abstractions layer

The *abstractions* layer defines a set of metamodel elements whose purpose is to represent domain-specific knowledge as well as to provide a business overview of legacy information systems. This layer has three packages: *structure*, *conceptual* and *build*; and each package defines its own model.

Throughout the modeling process, the *infrastructure* layer is not used to represent any specific model since the elements of this layer are used by other layers. The following KDM layers, *program elements* layer and *runtime resource* layer are used to model explicit knowledge of the legacy systems. Therefore the models built according to the packages of these layers can be obtained automatically through specific analytic tools. However, the last layer, the *abstractions* layer, is used when the KDM implementer find itself interpreting intent full-on. Thus, this layer focuses on representing the implicit knowledge of the legacy systems. For this purpose, the models built according to this layer are obtained by means of the analysis of the explicit knowledge of the models of the previous layers as well as the help of domain experts.

4.1.4.1. Structure package. The *structure* package represents the architectural components of legacy systems like subsystems, layers, packages and so on. This means that the *structure* package defines the divisions and subdivisions down to the modules defined in the *code* package.

The *structure* package defines the *StructureModel* element that represents the organization of the legacy source code, thus the *StructureModel* aggregates several structure metamodel elements. "Packages are the leaf elements of the structure model and they represent divisions of a system's code parts in discrete and non-overlapping manner" [19]. Then, each *Package* can be separated into *Subsystems*, *Layers* and *Components*, or *ArchitectureViews*.

4.1.4.2. Conceptual package. The *conceptual* package represents the domain-specific information of a legacy system in a conceptual model. The conceptual model can represent a behavior graph with the paths of the application logic and the associated conditions. This behavior graph is built from action elements in the program element layer, although the action elements are enriched with semantics of the domain. Therefore, a process of mining high-level knowledge from the previous low-level KDM models makes it possible to discover additional KDM entities and relationships that represent the *concepts* involved in the system. Nevertheless, the value-added knowledge discovery process is very difficult, and thus this process tends to involve domain experts and application experts.

The *ConceptualModel* element consists of an aggregation of *AbstractConceptual-Elements* (see Fig. 5). This element has a property named *implementation* which specifies the mapping of a specific concept to the *KDMEntities* in low-level models. The *AbstractConceptualElement* is specialized into three metamodel elements: the *ConceptualRole* element represents the role played by a participant in a conceptual association, the *TermUnit* represents a noun concept,

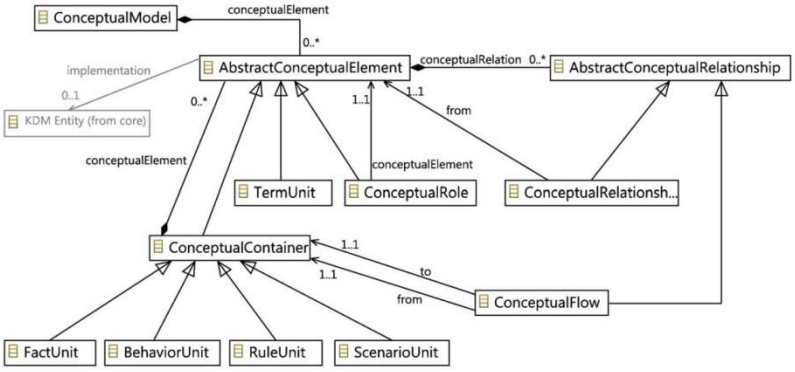


Fig. 5. The KDM conceptual model (adapted from [19]).

and the *ConceptualContainer* represents a container for conceptual entities.

Moreover, the *ConceptualContainer* element is also specialized into *FactUnit*, *RuleUnit*, *BehaviourUnit* and *ScenarioUnit* (see Fig. 5). *FactUnit* associates several conceptual elements and represents some behavior of the legacy system. The *RuleUnit* also associates multiple conceptual entities, but it specifies a condition or a constraint of the behavior of the legacy system. The *BehaviourUnit* element represents a behavior graph with several paths through the application logic. Finally, the *ScenarioUnit* element represents a single path or a set of paths through the behavior graph of the application logic. In addition, the *ConceptualContainer* elements are related by means of *ConceptualFlow* elements, the only kind of relationship defined by the *conceptual* package.

Also, it should be borne in mind that the conceptual package is aligned with the SBVR (*Semantics of Business Vocabulary and Business Rules*) specification [40]. SBVR is an OMG specification related to KDM, which defines a mechanism to discover and represent business rules from information systems. SBVR enables the storage of business rules and business vocabulary in general in an MOF-compliant repository. The KDM conceptual elements that represent the minimal units to represent concepts have a direct correspondence to the elements of SBVR specification: the *TermUnit* corresponds to *Noun* in the SBVR terminology that collectively refers to SBVR *Terms* and SBVR *Names*, the *FactUnit* of KDM corresponds to SBVR *Fact*, and *RuleUnit* corresponds to conditions or constraints in the SBVR specification.

4.1.4.3. Build package. The *build* package defines the artifacts related to the engineering view of a legacy system such as roles, development activities or deliverables generated by the build process. This package defines the *BuildModel* as a model for representing that information. The build model defines a set of build elements such as *Supplier* or *Tool*, and it also provides a set of build resources like *BuildProduct*, *BuildComponent*, *BuildStep*, *BuildDescription*, and so on.

4.2. KDM as an Ontology

KDM can be seen as an ontology [42], since KDM specifies a set of common concepts required for understanding legacy systems and provides infrastructure to support specialized definitions of domain-specific, application-specific and implementation-specific knowledge. Indeed, most of the KDM specification is a definition of a language- and platform-independent ontology of legacy information systems.

The underlying ontology defined in KDM provides the nomenclature needed to represent all the artifacts involved in legacy systems and their relationships. The ontology makes it possible to represent both the structure and behavior of a specific legacy information system at different abstraction levels due to the elements defined in the metamodel of the KDM standard.

With the KDM ontology, the deep semantic integration between software development tools that deal with legacy information systems can take place. At the integration level, the reverse engineering tools and analytic tools need to share the same ontology related to the application. For instance, consider the following three tools: firstly, a reverse engineering tool generates the code model from a source code file; secondly, an analytic tool can derivate the cyclomatic complexity metric of the same source file; and finally, another analytic tool specifies the business rules supported by that source file. These three tools must share the same ontology that involves the concept of *source file* [23].

4.2.1. Micro KDM

In addition, the KDM specification defines the *micro KDM* package in order to refine the semantic defined in the ontology. The *micro KDM* package is mainly applied to the action elements defined in the *action* package. *Micro KDM* is a compliance point for the *action* package of the *code* viewpoint. *Micro KDM* defines a set of compliance rules, additional guidance to building and interpreting extra high-fidelity KDM views, suitable to performing static analysis.

The *micro actions* defined by the catalog of the *micro KDM* package decorate the *macro actions* defined through the *action* elements of the *action* package. In this manner, the control flow and the data flow of the KDM representations are more precise [19].

Each micro action consists of four elements: (i) the action kind, specifying the nature of the operation performed, which is represented by means of the *kind* attribute of the macro action; (ii) outputs, representing the outgoing *Writes* relationships, which represent the result of the action; (iii) inputs, defining the outgoing *Reads* relationships, which represent the arguments of the micro action; and finally (iv) the control part, defining the outgoing control flow relationships.

The *micro KDM* package defines a wide catalog of micro actions, which is divided into ten categories that can be seen in Table 1.

4.3. KDM as a common interchange format

Each KDM representation is a model that represents a specific legacy system. However, a KDM representation is not a model that represents system constraints like a UML model used during the system design stage, but is an intermediate representation to capture

Table 1
Micro actions defined by the *micro KDM* package [19].

Category	Description	Micro Actions
Comparison	This category represents actions related to comparisons between value expressions.	Equals, NotEquals, LessThanOrEqual, LessThan, GreaterThan, GreaterThanOrEqual, Not, And, Or, Xor.
Numerical datatypes	This category represents micro actions related to the primitive numerical datatypes.	Add, Multiply, Negate, Subtract, Divide, Remainder, Successor.
Bitwise operations	This represents actions related to the bitwise operations on primitive datatypes.	BitAnd, BitOr, BitNot, BitXor, LeftShift, RightShift, BitRightShif.
Control	This category represents micro actions that represent the control flow.	Assign, Condition, Call, MethodCall, PtrCall, VirtualCall, Return, Nop, Goto, Throw, Incr, Decr, Swith, Compound.
Data Access	This represents actions related to data access.	FieldSelect, FieldReplace, ChoiceSelect, ChoiceReplace, Ptr, PtrSelect, PtrReplace, ArraySelect, ArrayReplace, MemberSelect, MemberReplace, New, NewArray.
Conversions	This represents the actions related to type conversions	Sizeof, Instanceof, DynCast, TypeCast.
String operations	This represents the actions related to string type operations.	IsEmpty, Head, Tail, Empty, Append.
Set type operations	This represents the actions related to set type operations. The value space of a set element is the set of all subsets of the value space of the element datatype.	IsIn, Subset, Difference, Union, Intersection, Select, IsEmpty, Empty.
Sequence type operations	This represents the actions related to the sequence type operations. The value of a sequence element is an ordered sequence of values of the element datatype.	IsEmpty, Head, Tail, Empty, Append.
Bag type operations	This represents the actions related to bag type operations. The value of a bag element is a collection of instances of values of the element datatype.	IsEmpty, Select, Delete, Empty, Insert, Serialize.

precise knowledge about the existing applications [19]. As a consequence, in addition to a metamodel and an ontology, the KDM specification can be seen as a common interchange format shared for reverse engineering tools, software analysis tools, and any other modernization tool.

The metamodel of the KDM standard defines a common repository structure to exchange the data contained within individual tool models that represent legacy software artifacts. This means that the metamodel defined by the KDM standard provides a common interchange format to enable the interoperability between knowledge discovery tools, services, and respective intermediate representations.

The KDM standard makes it possible to analyze artifacts of legacy information systems and export their representations based on the XMI (XML Metadata Interchange) format [19]. XMI is a model-driven XML (eXtensible Markup Language) integration framework for defining, manipulating and interchanging XML data and objects [38].

Fig. 6 shows an example of an XMI file obtained from a fragment of the legacy source code. The upper part of Fig. 6 represents a fragment of a Java source file that defines a package with two classes. The bottom part of Fig. 6 shows the XMI file that corresponds to the KDM representation of the Java source code. This KDM representation defines a KDM code model using the code and action packages, thus this KDM representation uses the *infrastructure* and *program element* layer. The code model uses the metamodel elements of the *code* package to represent the structure of the source code, i.e., the Java package, the two classes and their methods. In addition, this KDM representation employs the *action* metamodel elements to depict the meaning of each Java sentence.

4.3.1. KDM compliance levels

The metamodel of the KDM standard has a very broad scope since it makes it possible to represent legacy systems with different natures. However, not all KDM capabilities are always applicable to all platforms technologies, applications or programming languages.

Therefore, the KDM specification establishes a set of compliance levels. A compliance level is defined by convention and denotes a specific metamodel subset that ensures the interoperability of a certain kind of tool. Due to the fact that the metamodel of the KDM standard is structured modularly and follows the principle of separation of concerns, the KDM specification can easily define several compliance levels.

The KDM specification defines three compliance levels: level 0 (L0), level 1 (L1) and level 2 (L2).

- Level 0. This compliance level is formed by the following KDM packages: *core*, *kdm*, *source*, *code* and *action*, i.e., the packages of infrastructure layer and program element layer. A tool is L0 compliant when this tool supports all the model elements in those packages. Thus, the L0 compliance level represents the minimal basis for interoperability between different kinds of tools. In fact, to be L1 or L2 compliant, it is first necessary to be L0 compliant.
- Level 1. This compliance level is formed by the remaining KDM packages: *platform*, *data*, *event*, *ui*, *build*, structure and conceptual, which belong to the runtime resource layer and abstractions layer. Also, the elements defined in the *micro kdm* package are considered in this level. To be L1 level compliant, it is not necessary to support all the packages, rather only the packages for the specific domain addressed by the tool. This means that L1 represents a level where tools could be complementary since the tools can focus on different areas of concern.
- Level 2. This level is the union of all the KDM packages of the L1 level. Thus, in order to be L2 compliant, a tool should support the whole metamodel of the KDM standard.

Compliance levels allow tool vendors to select only those parts of the metamodel of the KDM standard that are interesting to a specific tool [19]. When a vendor starts to develop a software modernization tool, it knows *a priori* the parts of the metamodel of the KDM standard that should be supported by the tool in order to be in compliance with the desired level. As a consequence, all the tools in compliance with a specific level are interoperable between each other in that level.

4.3.2. KDM ecosystem

The KDM standard makes it possible to represent and interchange platform- and language-independent models about several artifacts of a legacy system. Due to this fact, the KDM specifications enable the automation and integration of the knowledge discovery processes, since it defines the items that a reverse engineering tool should discover and a software analysis tool can use. Therefore, the KDM standard facilitates knowledge-based integration between different tools.

The KDM standard changes the way in which reverse engineering tools are built and used. Firstly, the traditional reverse engineering

```
package flip;
public class foo {
    public foo() {
    }
    public flip(int i) {
        return i * -1;
    }
}
public class FlipClient {
        public static void main(String[] args) {
        foo f= new foo();
        iFlip g=(iFlip) f;
        f.flip(100);
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<kdm:Segment          xmlns:kdm="http://kdm.omg.org/kdm"          xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"       xmlns:action="http://kdm.omg.org/action"
xmlns:code="http://kdm.omg.org/code"                                                  xmi:version="2.0"
xsi:schemaLocation="http://kdm.omg.org/action
http://alarcos.esi.uclm.es/per/rpdelcastillo/metamodels/kdm.ecore.xml#//action
http://kdm.omg.org/code
http://alarcos.esi.uclm.es/per/rpdelcastillo/metamodels/kdm.ecore.xml#//code  http://kdm.omg.org/kdm
http://alarcos.esi.uclm.es/per/rpdelcastillo/metamodels/kdm.ecore.xml#//kdm"
name="InterfaceExample.java.kdm">
  <model xsi:type="code:CodeModel" xmi:id="id.0" name="InterfaceExample.java">
    <codeElement xsi:type="code:LanguageUnit" xmi:id="id.1" name="Common Java Datatypes" />
    <codeElement xsi:type="code:Package" xmi:id="id.2" name="flip">
      <codeElement xsi:type="code:ClassUnit" xmi:id="id.3" name="foo">
        <codeElement xsi:type="code:Signature" xmi:id="id.4" name="foo" />
        <codeElement xsi:type="code:MethodUnit" xmi:id="id.5" kind="constructor" name="foo">
          <codeElement xsi:type="code:Signature" xmi:id="id.6" name="foo" />
        </codeElement>
        <codeElement xsi:type="code:MethodUnit" xmi:id="id.7" kind="abstract" name="flip">
          <codeElement xsi:type="code:Signature" xmi:id="id.8" name="flip">
            <parameterUnit xmi:id="id.9" name="i" />
          </codeElement>
        </codeElement>
      </codeElement>
      <codeElement xsi:type="code:ClassUnit" xmi:id="id.10" name="FlipClient">
        <codeElement xsi:type="code:Signature" xmi:id="id.11" name="FlipClient" />
        <codeElement xsi:type="code:MethodUnit" xmi:id="id.12" name="main">
          <codeElement xsi:type="code:Signature" xmi:id="id.13" name="main">
            <parameterUnit xmi:id="id.14" name="args" />
          </codeElement>
          <codeElement xmi:id="id.15" xsi:type="code:StorableUnit" name="f" kind="local" />
          <entryFlow xmi:id="id.16" from="id.12" to="id.17" />
          <codeElement xsi:type="action:ActionElement" xmi:id="id.17" name="a1" kind="Assign">
            <actionRelation xsi:type="action:Reads" xmi:id="id.18" from="id.17" to="id.15" />
            <actionRelation xsi:type="action:Flow" xmi:id="id.19" from="id.17" to="id.22" />
          </codeElement>
          <codeElement xmi:id="id.20" xsi:type="code:StorableUnit" name="g" kind="local" />
          <codeElement xsi:type="action:ActionElement" xmi:id="id.22" name="a2" kind="MethodCall"/>
        </codeElement>
      </codeElement>
    </codeElement>
  </model>
</kdm:Segment>
```

Fig. 6. Example of KDM representation based on XMI from a fragment of Java source code.

tools have been built as silos[1] where each tool focuses on recovering and analyzing different *proprietary* contents in an isolated way (see Fig. 7 left side). For instance, suppose that we carry out a knowledge discovery process of a legacy enterprise application. We would use a reverse engineering tool for the source code and another tool for the legacy database. As a consequence, at the end of the process we would have two proprietary and independent models: a source code model and a database model. These models must be also analyzed independently through two different analytic tools because the current elements do not match in their representations.

Secondly, the KDM standard makes it possible to build reverse engineering tools in a KDM ecosystem. This means that the reverse engineering tools can recover different knowledge related to different artifacts, but this knowledge is represented and managed in an integrated and standardized manner through KDM (see Fig. 7 right side). In this way, the analytic tools can analyze the shared KDM repository and generate the new knowledge. Thus, the advantages derived from the integration of this kind of tool are the following:

- It makes it possible to deal with a diversity of implementation languages and runtime platforms.
- In the future, more analytic tools can be plugged into the KDM models homogeneously to generate more and more valuable knowledge. Thus, it facilitates the complete integration of the different views of the legacy system.
- It makes it possible to understand the "as is" architecture of the legacy system and manages changes from an architectural perspective.
- It makes it possible to combine static analysis with dynamic analysis and metrics for a great amount of software artifacts.

5. Using KDM to modernize legacy systems

The software modernization approach, and particularly ADM, is a mechanism to deal with negative software erosion effects in legacy

[1] Silo: in the context of KDM, a silo is a software solution that consists of several applications used in a sequential way to reach a common goal. The applications of the solution use a common but proprietary format to interchange the shared knowledge throughout the silo. Thus, other similar applications cannot be integrated beyond the solution.

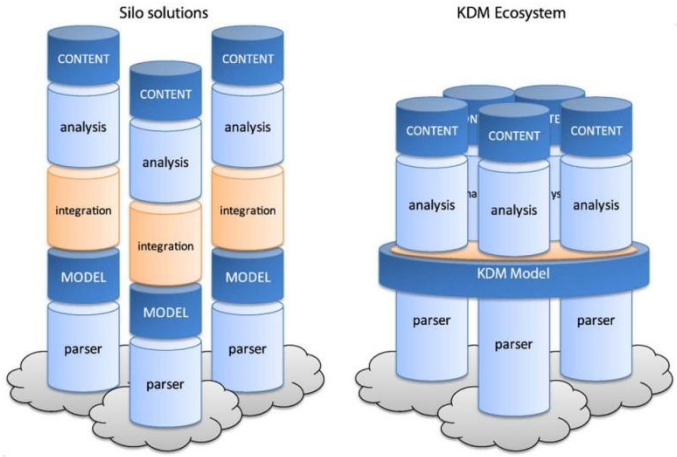


Fig. 7. Software archeology tools: silo solutions (left side) and KDM ecosystem (right side).

systems. According to [35], ADM is the concept of modernizing existing systems with a focus on all aspects of the current system architecture and the ability to transform current architectures to target architectures. Thus, an ADM-based process understands and evolves existing software assets and restores the value of existing applications.

ADM is based on traditional reengineering processes, but it takes *model-driven* principles into account. The horseshoe reengineering model has been adapted to ADM and it is known as the horseshoe modernization model [24]. This model is divided into three stages (see Fig. 8):

- Reverse engineering represents the left side of the horseshoe. It analyzes the legacy system in order to identify the components of the system and their interrelationships. In turn, the reverse engineering stage builds one or more representations of the legacy system at a higher level of abstraction.
- Restructuring represents the curve of the horseshoe since this stage takes the previous system's representation and transforms it into another at the same abstraction level. This stage preserves the external behavior of the legacy system.
- Forward engineering represents the right side of the horseshoe because it generates physical implementations of the target system at a low abstraction level from the previous restructured representation of the system.

KDM make it possible to represent the legacy knowledge extracted from different software artifacts in legacy systems. The software artifacts are analyzed and the KDM models can be built directly according to the metamodel of the KDM standard. However, the KDM models can be also built in two steps: (i) firstly, the recovered information from the software artifacts is used to obtain a PSM model according to a specific metamodel (for example, a code model according to a Java metamodel); and then, (ii) the KDM model, which can be considered as a PIM model, is obtained from the earlier specific model by means of model transformations.

From the point of view of KDM, the reverse engineering stage is probably the most important stage in the horseshoe modernization process. This is due to the fact that this activity conditions the abstraction level achieved in each kind of modernization process and therefore the resources and possibilities to restructure after the legacy systems. The reverse engineering stage is the main activity especially

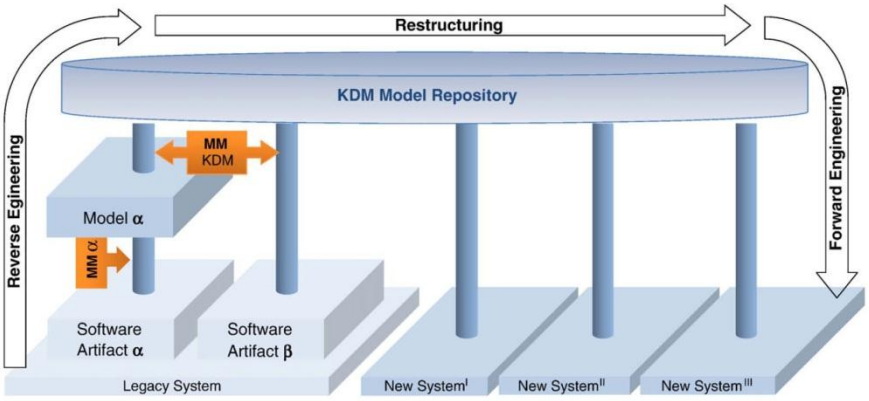


Fig. 8. The role of KDM within horseshoe modernization model.

when a specific modernization process wishes to obtain business process models, business rules and any other business semantics from the legacy systems, i.e., the higher abstraction level. In this kind of modernization process there is a large conceptual gap between business processes and legacy systems that needs to be gradually reduced. KDM reduces this gap due to the fact that it is organized into several layers at different abstraction levels. Thus, KDM makes it possible to carry out *endogenous transformations*[2] from models in lower layers to models in higher layers of the KDM structure. Therefore, specific business knowledge can be extracted directly from the legacy system, and then the implicit knowledge at higher abstraction levels is inferred or deduced from the prior knowledge.

KDM also assists in the restructuring and forward stages of the modernization process. In this case, KDM acts as common interchange format for several restructuring tools that can modify, restructure and improve the recovered KDM models. In addition, KDM offers a KDM model repository according to the KDM ecosystem. Thus, the forward engineering stage can be aided by KDM because the KDM model repository is also shared by forward engineering tools such as generative tools. Automatic Code Generation tools can use KDM models at the begining of the modernization process to generate the respective source code and other software artifacts regarding the target information system by using forward engineering techniques.

5.1. ADM standards

ADM not only defines KDM, but also defines an entire set of standards that complement the KDM standard in order to support all of the throughout the whole ADM-based process. The other standards are being developed around KDM, although some of them are still in the approval or development stage [43]:

- ASTM (Abstract Syntax Tree Metamodel) represents knowledge discovered in legacy systems at the intermediate level of abstraction under KDM by means of abstract syntax tree models.
- SMM (Software Metrics Package) defines a metamodel for representing measurement information related to software, its operation, and its design.
- ADM Pattern Recognition specification aims to be a standard to facilitate the examination of structural metadata with the intent of deriving patterns about legacy systems. It is an on-going activity on the roadmap of the ADM Task Force, leading to a request for proposals in the future.
- ADM Visualization specification focuses on the different ways to show application metadata stored within the KDM models. It is an on-going activity on the roadmap of the ADM Task Force.
- ADM Refactoring specification defines ways in which the KDM specification can be used to refactor applications by means of structuring, rationalizing, modularizing, and so on. It is also an on-going activity on the roadmap of the ADM Task Force.
- ADM Transformation specification seeks to define mappings between KDM, ASTM and target models. It is an on-going activity on the roadmap of the ADM Task Force.

Fig. 9 shows the set of ADM standards contextualized in the horseshoe modernization model as well as other OMG business modeling standards related to ADM-based processes. The KDM standard (ISO/IEC 19506) is the core for the rest of the standards, although KDM is itself sufficient to address software modernization processes that minimize the negative effects of software erosion.

5.2. Using ADM/KDM in research and industrial projects

The KDM standard and the ADM paradigm in general have been applied in several software modernization scenarios. Two important European projects, *MOMOCS* [29] and *ModelWare* [27], aim to lead the successful adoption of ADM by the industry. While the *ModelWare* project focuses on most parts of model-driven development, the *MOMOCS* project particularly addresses the activities and techniques of the software modernization approach.

In addition, there are several companies that use the KDM standard. *KDM Analytics* was the first company to provide a reference implementation of KDM known as *KDM SDK 2.0* [22]. This implementation is an *Eclipse™* EMF plug-in which provides a set of tools for working with KDM. *KDM SDK 2.0* is an adoption kit that facilitates the design of mappings from proprietary internal representation in the metamodel of the KDM standard and jumpstarts development of KDM tools. In 2009, *KDM Analytics* provided the *KDM Workbench* tool, which has *Eclipse™* plug-ins for C, C++, Java, and Cobol.

Other companies have also used KDM in some projects. For example, Benchmark Consulting Services Inc. addresses the modernization of mainframe Cobol systems using KDM [6]. Adaptive Inc. also uses the KDM standard together with other OMG standards in order to evolve legacy information systems [2]. MicroFocus provides a bridge from their software modernization tool for Cobol, SQL and CICS to KDM [26]. Allen Systems Group Inc. proposes integrated modernization environment (IME) supported by a multi-technology repository based on KDM [3]. *Government Computer News* presents a technical report that shows why organizations that provide tools and services for software development lifecycle may need KDM [14].

MoDisco is another important tool that consists of an *Eclipse™* plug-in for model-driven reverse engineering. With *MoDisco*, the practical extraction of models can be done from various kinds of legacy systems. In addition, *MoDisco* proposes a generic and extensible metamodel-driven approach to model discovery [28].

Moreover, *MARBLE*, a non-commercial tool based on the *Eclipse™* platform, can also be used to recover business processes from legacy systems in order to carry out business modernization processes [45]. This tool obtains business processes through three transformations: firstly, the tool recovers PSM models from different legacy software artifacts; secondly, the tool integrates those models into a single PIM model according to the metamodel of the KDM standard; and finally, *MARBLE* recovers a business process model by applying a set of business patterns in the KDM model [46]. MARBLE can be seen as a set of tools working against the same KDM repository.

Ulrich and Newcomb have collected various ADM-based case studies [50]. Some of them consist of real world case studies involving the automated transformation of information systems, from their legacy source code, into modernized languages and platforms in which KDM is used to represent and manage all this information. For example, DelaPeyronnie et al. [12] present an ADM-based project for modernizing the EATMS system, an air traffic management system used in 280 airports worldwide. This project was carried out in a platform migration scenario, since the main objective was the transformation of the legacy system from the *Ada* language into the high-performance, real-time *Java* language. Another successful case study was the modernization project presented by Barbier et al. [5], which focuses on how PIM models can be automatically generated from Cobol-based legacy systems.

Finally, KDM is being used in a wide variety of projects in the software assurance field, as a vendor-neutral standard to formalize machine-readable content for assurance, vulnerability patterns, to perform static analysis directly on the KDM representation of code, as

[2] Endogenous transformation. Model transformations can be classified into endogenous and exogenous transformations. While exogenous transformations involve two models represented according to two different metamodels, endogenous transformations are performed between two models represented according to the same metamodel.

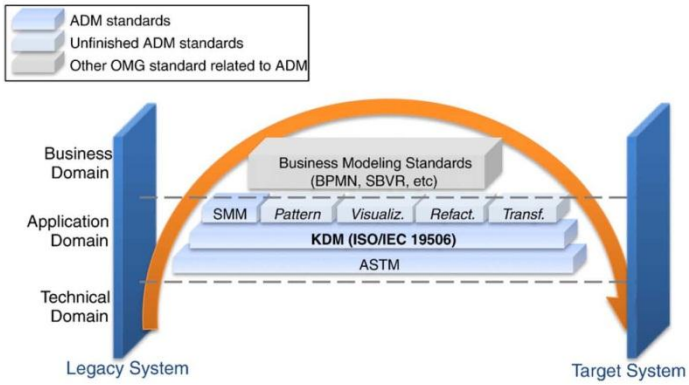


Fig. 9. OMG ADM Standards within the horseshoe modernization model.

well as for automatically generating test cases in multiple programming languages for the security vulnerability detection tools.

6. Conclusions

ADM advocates carrying out reengineering processes takes MDA principles into account. ADM enables the evolution of the legacy systems eradicating (or at least minimizing) the negative effects of the software erosion phenomenon. The main advantage of ADM with respect to traditional evolutionary maintenance processes like reengineering is that it formalizes and standardizes evolutionary maintenance processes. With this advantage, the modernization process can be automatic, reusable and repeatable.

This paper has shown the KDM standard that belongs to the set of standards to modernize legacy information systems defined by OMG within the ADM Task Force. KDM is the first standard defined by the ADM Task Force and it is also the first standard ratified by the ISO/IEC organization as ISO/IEC 19506. The KDM standard makes it possible to represent and manage the knowledge recovered throughout all the modernization stages: reverse engineering, restructuring and forward engineering.

The KDM standard defines a metamodel that specifies a broad set of metamodel elements depicting all the different software artifacts involved in legacy information systems. The metamodel defined by the KDM standard is organized into four abstraction layers, and thus it makes the representation of different models possible that depict knowledge embedded in legacy artifacts at different abstraction levels.

Furthermore, KDM can be seen as a standard that defines an ontology that includes all the elements involved in legacy systems. KDM defines the *micro KDM* package to refine the semantic of some metamodel elements defined in the metamodel of the KDM standard.

The underlying ontology together with the metamodel defined by the KDM standard provides a common semantic for artifacts of several kinds of legacy systems. As a consequence, the KDM standard can be also used as a common interchange format by modernization tools. The different KDM models can be represented as XMI files with a well-defined format according to the metamodel of the KDM standard. These files share the information of the KDM models between different tools. For instance, some reverse engineering tools can recover KDM models from the legacy systems, and then, other analytic or refactoring tools can use those KDM models.

Therefore, the KDM standard solves the four challenges introduced at the beginning of this paper about knowledge management in software modernization processes:

- The layer organization of the metamodel of the KDM standard enables the knowledge representation of all the software artifacts (source code, databases, user interfaces, business rules, etc.) from different viewpoints and different abstraction levels.
- In addition, the KDM standard makes it possible to manage the legacy knowledge recovered from a legacy system throughout all the stages in an ADM-based process due to the fact that ADM provides a set of complementary standards.
- The knowledge recovered from legacy systems and represented in KDM models can be shared by different modernization tools according to the KDM ecosystem.
- Finally, due to the fact that KDM is divided into different abstraction layers, it is able to establish endogenous model transformations between layers that represent the explicit knowledge and layers that represent the implicit knowledge of legacy systems.

Acknowledgments

This work was supported by the Spanish FPU Programme, and by the R+D projects ALTAMIRA (JCCM, PII2I09-0106-2463), PEGASO/MAGO (MICIN and FEDER, TIN2009-13718-C02-01) and MEDUSAS (CDTI (MICINN), IDI-20090557).

References

[1] M.S. Abdullah, et al., Knowledge modelling using uml profile for knowledge-based systems development, in Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering, Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies, IOS Press, 2007, pp. 74–89.
[2] Adaptive Inc. *Adaptive standards* 2008, [cited 2009 18/12/2009]; Available from: http://www.adaptive.com/standards.html.
[3] Allen Systems Group I, Presentation to Address Integrated Modernization Environment (IME) Issues and Market-Leading ASG-becubic™ Multi-Technology Repository, 2009, [cited 2009 18/12/2009]; Available from: http://www.asg.com/newsroom/pr_details.asp?id=136.
[4] F. Armour, et al., A UML-driven enterprise architecture case study, Proceedings of the 36th Annual Hawaii International Conference on System Sciences, IEEE Computer Society, 2003.
[5] F. Barbier, et al., Model-driven reverse engineering of COBOL-based applications, in: W.M. Ulrich, P.H. Newcomb (Eds.), Information Systems Transformation. Architecture Driven Modernization Case Studies, Morgan Kauffman, Burlington, MA, 2010, pp. 283–299.

[6] Benchmark Consulting Services I, Mainframe COBOL Application Modernization: A Revolutionary Approach to Evolving Legacy System. [cited 2009 18/12/2009]; Available from, http://www.benchmarkconsulting.com/AboutUs_nr_ct.html 2007.
[7] K.H. Bennett, V.T. Rajlich, Software maintenance and evolution: a roadmap, Proceedings of the Conference on the Future of Software Engineering, ACM, Limerick, Ireland, 2000.
[8] A. Bianchi, et al., Iterative reengineering of legacy systems, IEEE Transactions on Software Engineering 29 (3) (2003) 225–241.
[9] G. Canfora, M. Di Penta, New frontiers of reverse engineering, 2007 Future of Software Engineering, IEEE Computer Society, 2007.
[10] E.J. Chikofsky, J.H. Cross, Reverse engineering and design recovery: a taxonomy, IEEE Software 7 (1) (1990) 13–17.
[11] A. Daga, et al., An ontological approach for recovering legacy business content, Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05), IEEE Computer Society, 2005, pp. 224–232, Track 8 - Volume 08.
[12] J. DelaPeyronnie, et al., Modernization of the Eurocat Air Traffic Management System (EATMS), in: W.M. Ulrich, P.H. Newcomb (Eds.), Information Systems Transformation. Architecture Driven Modernization Case Studies, Morgan Kauffman, Burlington, MA, 2010, pp. 91–131.
[13] L. Favre, Modernizing software & system engineering processes, International Conference on Systems Engineering, IEEE Computer Society, 2008.
[14] Government Computer News, All for one, but not one for all, 2007. Mar 18, 2007; Available from: http://gcn.com/articles/2007/03/18/all-for-one-but-not-one-for-all.aspx.
[15] G. Grau, X. Franch, N.A.M. Maiden, PRiM: An i*-based process reengineering method for information systems specification, Inf. Softw. Technol. 50 (1–2) (2008) 76–100.
[16] ISO/IEC, ISO/IEC 9496:2003, CHILL - The ITU-T programming language, 2003, http://www.iso.org/iso/catalogue_detail.htm?csnumber=34084, 2003, ISO/IEC. p. 221.
[17] ISO/IEC, ISO/IEC 14764:2006, Software engineering—software life cycle processes—maintenance http://www.iso.org/iso/catalogue_detail.htm?csnumber=39064 2006, ISO/IEC.
[18] ISO/IEC, ISO/IEC 11404:2007, General-Purpose Datatypes (GPD), http://www.iso.org/iso/catalogue_detail.htm?csnumber=39479, 2007, ISO/IEC. p. 96.
[19] ISO/IEC, ISO/IEC DIS 19506. Knowledge Discovery Meta-model (KDM), v1.1 (Architecture-Driven Modernization), http://www.iso.org/iso/catalogue_detail.1128htm?csnumber=32625, 2009, ISO/IEC. p. 302.
[20] J. Jokinen, H.-M. Järvinen, T. Mikkonen, Incremental introduction of behaviors with static software architecture, Comput. Stand. Interfaces 25 (3) (2003) 215–222.
[21] R. Kazman, S.G. Woods, S.J. Carrière, Requirements for integrating software architecture and reengineering models: CORUM II, Proceedings of the Working Conference on Reverse Engineering (WCRE'98), IEEE Computer Society, 1998.
[22] KDM Analytics, *Knowledge Discovery Metamodel (KDM) Software Development Kit 2.0 Eclipse Plugin*KDM Analytics, Inc, http://www.kdmanalytics.com/kdmsdk/KDMSDK_brochure.pdf 2008.
[23] KDM Analytics, *Why Knowledge Discovery Metamodel?* KDM Analytics, Inc, 2008, http://kdmanalytics.com/kdm/ why_kdm.php.
[24] V. Khusidman, W. Ulrich, Architecture-Driven Modernization: Transforming the enterprise. DRAFT V.5 http://www.omg.org/docs/admtf/07-12-01.pdf 2007. OMG. p. 7.
[25] L.A. Maciaszek, Roundtrip architectural modeling, Proceedings of the 2nd Asia-Pacific Conference on Conceptual modelling, 43, Australian Computer Society, Inc, Newcastle, New South Wales, Australia, 2005, pp. 17–23.
[26] Micro Focus, Modernization Workbench™, http://www.microfocus.com/products/modernizationworkbench/ 2009.
[27] ModelWare. MODELWARE. Project co-funded by the European Commission under the "Information Society Technologies" Sixth Framework Programme (2002–2006) 2006 08-2009]; Available from: http://www.modelware-ist.org/.
[28] MoDisco, KDM-to-UML2 converter, http://www.eclipse.org/gmt/modisco/toolBox/KDMtoUML2Converter/.
MoDisco, Eclipse Incubation Project. IST European MODELPLEX project (MODELing solution for comPLEX software systems, FP6-IP 34081), 2008.
[29] MOMOCS, MOdel driven MOdernisation of Complex Systems is an EU-Project, http://www.momocs.org/ 2008; [cited 2008; Available from: http://www.momocs.org/.
[30] B. Moyer, Software archeology. Modernizing old systems, Embedded Technol. J. 1 (2009) 1–4.
[31] M.z. Muehlen, M. Indulska, G. Kamp, Business process and business rule modeling languages for compliance management: a representational analysis, Tutorials, Posters, Panels and Industrial Contributions at the 26th International Conference on Conceptual Modeling, 83, Australian Computer Society, Inc, Auckland, New Zealand, 2007, pp. 127–132.
[32] M.L. Nelson, R.L. Rariden, R. Sen, A lifecycle approach towards business rules management, International Conference on System Sciences, IEEE Computer Society, Hawaii, 2008.
[33] P. Newcomb, Architecture-Driven Modernization (ADM), Proceedings of the 12th Working Conference on Reverse Engineering, IEEE Computer Society, 2005.
[34] OMG, Why Do We Need Standards for the Modernization of Existing Systems?, 2003, OMG ADM Task Force.
[35] OMG, ADM Glossary of Definitions and Terms, http://adm.omg.org/ADM_Glossary_Spreadsheet_pdf.pdf 2006, OMG. p. 34.
[36] OMG, Architecture-Driven Modernization Roadmap, Object Management Group, 2006.
[37] OMG, Meta Object Facility (MOF™) Version 2.0 http://www.omg.org/spec/MOF/2.0/PDF/ 2006, OMG.
[38] OMG, XML Metadata Interchange. MOF 2.0/XMI Mapping, v2.1.1 http://www.omg.org/spec/XMI/2.1.1/PDF 2007, OMG.
[39] OMG, QVT. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification http://www.omg.org/spec/QVT/1.0/PDF 2008, OMG.
[40] OMG, Semantics of Business Vocabulary and Business Rules (SBVR), v1.0. OMG Available Specification http://www.omg.org/docs/formal/08-01-02.pdf, 2008, OMG p. 392.
[41] OMG, SPEM. Software & Systems Process Engineering Meta-Model, v 2.0 Specification http://www.omg.org/cgi-bin/doc?formal/2008-04-01 2008, OMG.
[42] OMG, Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM), v1.1, http://www.omg.org/spec/KDM/1.1/PDF/2009, OMG. p. 308.
[43] OMG, Architecture-Driven Modernization Standards Roadmap. 2009 February 2009 29/10/2009]; Available from:http://adm.omg.org/ADMTF%20Roadmap.pdf.
[44] B. Paradauskas, A. Laurikaitis, Business knowledge extraction from legacy information systems, Journal of Information Technology and Control 35 (3) (2006) 214–221.
[45] R. Pérez-Castillo, et al., MARBLE: A Modernization Approach for Recovering Business Processes from Legacy Systems, International Workshop on Reverse Engineering Models from Software Artifacts (REM'09), Simula Research Laboratory Reports, Lille, France, 2009, pp. 17–20.
[46] R. Pérez-Castillo, et al., Business process patterns for software archeology, 25th Annual ACM Symposium on Applied Computing (SAC'10), ACM, Sierre, Switzerland, 2010, pp. 165–166.
[47] R. Pérez-Castillo, et al., PRECISO: a reengineering process and a tool for database modernisation through web services in 24th Annual ACM Symposium on Applied Computing (SAC'09), 2009, pp. 2126–2133, Hawaii, USA.
[48] H.M. Sneed, Estimating the costs of a reengineering project, Proceedings of the 12th Working Conference on Reverse Engineering, IEEE Computer Society, 2005, pp. 111–119.
[49] W.M. Ulrich, Legacy Systems: Transformation Strategies, Prentice Hall, 2002, p. 448.
[50] W.M. Ulrich, P.H. Newcomb, Information systems transformation, Architecture Driven Modernization Case Studies, Morgan Kauffman, Burlington, MA, 2010, p. 429.
[51] J. Vara, et al., Supporting model-driven development of object-relational database schemas: a case study, in: R. Paige (Ed.), Theory and Practice of Model Transformations, Heidelberg, Springer Berlin, 2009, pp. 181–196.
[52] G. Visaggio, Ageing of a data-intensive legacy system: symptoms and remedies, Journal of Software Maintenance 13 (5) (2001) 281–308.
[53] H.-L. Yang, H.-C. Ho, Emergent standard of knowledge management: hybrid peer-to-peer knowledge management, Journal of Computer Standards and Interfaces 29 (4) (2007) 413–422.
[54] You, E., Modelling Strategic Relationships for Process Reengineering, PhD. thesis. 1995, University of Toronto.
[55] Y. Zou, et al., Model-Driven Business Process Recovery, Proceedings of the 11th Working Conference on Reverse Engineering (WCRE 2004), IEEE Computer Society, 2004, pp. 224–233.

Ricardo Pérez-Castillo holds the MSc degree in Computer Science from the University of Castilla-La Mancha, and he is currently a PhD student in Computer Science. He Works in Alarcos Research Group at the University of Castilla-La Mancha. His research intererests include architecture-driven modernization, model-driven development and business process recovery. Contact him at Escuela Superior de Informática, Paseo de la Universidad 4, 13071-Ciudad Real, Spain; Ricardo.PdelCastillo@uclm.es.

Ignacio García-Rodriguez de Guzmán is assistant professor at the University of Castilla-La Mancha and belongs to the Alarcos Research Group at the UCLM. He holds the PhD degree in Computer Science from the University of Castilla-La Mancha. His research interests include software maintenance, software modernization and service-oriented architecture. Contact him at Escuela Superior de Informática, Paseo de la Universidad 4, 13071-Ciudad Real, Spain; Ignacio.GRodriguez@uclm.es.

Mario Piattini is full professor at the UCLM. His research interests include software quality, metrics and maintenance. He holds the PhD degree in Computer Science from the Technical University of Madrid, and leads the Alarcos Research Group at the Universidad de Castilla-La Mancha. He is CISA, CISM e CGEIT by ISACA. Contact him at Escuela Superior de Informática, Paseo de la Universidad 4, 13071-Ciudad Real, Spain; Mario.Piattini@uclm.es.

2.3. Survey of Modernization Tools

The last section of the chapter concerning the state-of-the-art presents a brief survey of common current software modernization tools. It analyzes eleven tools by comparing their main functionalities. The survey also analyzes whether or not the tools implement the model-driven development principles in order to address the formalization and automation of large scale modernization projects.

The content of this section corresponds to a paper published in the IEEE Software Magazine. This journal had an impact factor of 1.511 in 2010, according to Thomson's Science Citation Index (SCI). The reference of this paper is: *Pérez-Castillo, R., I.G.-R. deGuzmán, M. Piattini, and C. Ebert, Reengineering Technologies. IEEE Software Magazine, 2011. 28(6): p. 13-17.*

SOFTWARE TECHNOLOGY

Editor: Christof Ebert
Vector Consulting Services
christof.ebert@vector.com

Reengineering Technologies

Ricardo Pérez-Castillo, Ignacio García-Rodriguez de Guzmán, Mario Piattini, and Christof Ebert

Software systems must continually evolve to meet ever-changing needs. However, such systems often become legacy systems as a consequence of uncontrolled maintenance combined with obsolete technology. To control maintenance costs and preserve complex embedded business rules, companies must evolve their legacy systems. This article introduces technologies for software reengineering. I look forward to hearing from both readers and prospective authors about this column and the technologies you want to know more about. —*Christof Ebert*

REENGINEERING IS THE most widely used approach to support evolutionary maintenance. It examines and alters a legacy system to reconstitute it in a new form.[1]

Reengingeering Stages

Reengineering consists of three stages: reverse engineering, restructuring, and forward engineering.

Reverse Engineering

Maintainers use reverse engineering to obtain one or more abstract representations of the legacy systems—for example, a set of class diagrams representing the system design—that identify the system's components and their interrelationships at a higher abstraction level.[2] Reverse engineering techniques fit into two categories: static and dynamic analysis.[3]

Static analysis syntactically analyzes the source code's structure to highlight weaknesses, complexities, and opportunities for improvement. Maintainers can also apply static analysis to database scripts to determine the database design.

Dynamic analysis examines legacy systems at runtime. It instruments source code to enable the code to register traces during execution. It then analyzes those traces to retrieve abstract representations of the legacy system. It can retrieve particular information—for example, performance data and values of data variables—that's known only during system execution.

> Reengineering consists of three stages: reverse engineering, restructuring, and forward engineering.

Restructuring

Restructuring transforms the abstract representations that reverse engineering provides into other representations at the same abstraction level. It aims to improve certain properties such as the representation's structure or quality as well as to introduce new business requirements. For example, maintainers can use restructuring to apply design patterns to maintain the system design. Restructuring techniques include *refactoring* to restructure source code or a database schemas by preserving its external behavior, *clone detection* to detect duplicate fragments of source code, and *dead-code detection* for removing inoperative code.

Forward Engineering

Maintainers use the restructured representations of legacy systems to generate physical implementations of the target system at a low abstraction level. Forward engineering involves using generative and transformational techniques to automatically obtain source code according to new platforms or programming languages.

0740-7459/11/$26.00 © 2011 IEEE

TABLE 1

Software reengineering tools.

Tool	Type	Reengineering stages supported	Techniques supported	Model-driven	Input artifacts	Output artifacts	Usefulness for maintainers	Pricing (US$)
Amelio	Commercial	All	Static analysis, refactoring, and automatic code generation	No	Source code and relational databases	Data-flow and process-flow graphs, along with source code	Reliable planning, calculation, and accomplishment of modernization projects	$0.42–$2.10 per LOC, depending on system complexity
Avalanche	Academic	Reverse engineering	Dynamic analysis	No	Binary code	Control and data-flow graphs and metrics	Program comprehension and detection of bugs and vulnerabilities	Free (open source code)
Blue Age	Commercial	All	Static analysis, refactoring, and automatic code generation	Yes	Source code (various languages)	System design diagrams and source code (different languages)	Migrates and transforms applications	n/a
CloneDR	Commercial	Reverse engineering and restructuring	Static analysis and clone detection	No	Source code (various languages)	Removing clone code	Maintainability improvement by removing duplicate code	$0.005–$0.03 per LOC
DB-MAIN	Academic	All	Static analysis, refactoring, and code generation	No	Relational and non-relational databases and database scripts	Database design, database scripts, and source code	Migration, integration, and federation of legacy databases; impact analysis; and data wrapper design and generation	Free
Microfocus Modernization Workbench	Commercial	Reverse engineering and restructuring	Static analysis	No	Application portfolio	Business knowledge	Locates where changes should be made to an application without disrupting stability	n/a

Model-Driven Reengineering

Software engineers have applied model-driven development (MDD) technologies to software reengineering to deal with standardization and automation challenges. In 2007, the Object Management Group (OMG) launched the Architecture-Driven Modernization (ADM) initiative to standardize model-driven reengineering. ADM advocates following MDD principles on reengineering projects, such as treating all the artifacts involved as models and providing model transformation among them. These let maintainers carry out more automatic and reusable reengineering projects because models are based on standard metamodels and model transformations are implemented using purpose-specific languages.

Maintainers adapt reengineering techniques to manage models using model refactoring rather than traditional refactoring. They also use static and dynamic analyses to generate models with the information retrieved according to metamodels, and so on.

The ADM initiative has also created the Knowledge Discovery Metamodel (KDM), which has been recently adopted

TABLE 1 (CONT'D)

Software reengineering tools.

Tool	Type	Reengineering stages supported	Techniques supported	Model-driven	Input artifacts	Output artifacts	Usefulness for maintainers	Pricing (US$)
MoDisco	Academic	Reverse engineering and restructuring	Static analysis and refactoring	Yes	Source code, databases and other artifacts	Models representing artifacts	A generic and extensible metamodel-driven approach to model discovery	Free (Eclipse plug-in)
Moose	Academic	Reverse engineering and restructuring	Static analysis, refactoring, and clone detection	Yes	Source code and user interfaces	System design diagrams	Allows understanding of legacy systems and evaluating their maintainability	Free (open source code)
Obeo Agility	Commercial	All	Static analysis, refactoring, and automatic code generation	Yes	Source code (various languages)	System design diagrams and source code (different languages)	Migrates and transforms applications	n/a
Resource Miner	Commercial	Reverse engineering and restructuring	Static analysis and refactoring	No	Source code (various languages)	Code inventory, requirements, and quality audits	Software asset management and application measure	$999 single license per year
Visual Paradigm	An integrated development environment supporting reengineering	Reverse engineering and forward engineering	Static analysis and automatic code generation	No but based on the Unified Modeling Language (UML)	Java, C++ java code or UML class diagrams	UML class diagrams, Java, or C++ code	Round-trip between Java or C++ and system design based on UML class diagrams	$699 single license and $67,000 corporate license (free community edition)

as the ISO/IEC 19506 standard.[4] KDM is the cornerstone of the ADM initiative and provides a common format for the interchange of information between reengineering tools. KDM facilitates the representation of information retrieved by reverse engineering from different legacy software artifacts and system viewpoints (for example, source code, database, and user interfaces). KDM is comparable to the UML standard. Whereas UML is used to generate new code in a top-down manner, a reengineering process involving KDM starts from the existing code and builds higher-level models in a bottom-up manner.

KDM-based model-driven reengineering alters the construction and use of reengineering tools. Traditional reengineering tools have been built as silos from which each tool recovers and analyzes a particular proprietary content in a single silo—for example, one tool would be used for the source code (such as Cobol), and another would be used for the legacy database (such as CICS/DB2; Customer Information Control System/Database 2). So, two proprietary and independent models exist at the end of the process—a Java code model and a DB2 database schema model—which should also be analyzed independently.

ADM encourages maintainers to build reengineering tools in a KDM ecosystem. This means that reengineering tools recover different information related to different legacy software artifacts and that a KDM repository is progressively populated with the information retrieved. Software analysis tools can be homogeneously plugged into the KDM common repository to generate more valuable information in a standard way.

Software Reengineering Tools

Table 1 compares 13 tools that support reverse engineering and reengineering.

SOFTWARE TECHNOLOGY

REENGINERING WITH MODISCO

MoDisco (http://eclipse.org/MoDisco) supports model-driven reengineering and focuses on reverse engineering and restructuring. MoDisco facilitates modernization of existing systems by providing

- metamodels to describe existing systems—for example, the Knowledge Discovery Metamodel (KDM) or Unified Modeling Language (UML),
- discoverers to automatically create models of existing systems—for example, from Java, C, or PHP projects, and
- generic tools to understand and transform complex models created from the existing systems—for example, graphical UML editors.

Figure A shows a reengineering project with MoDisco aimed at evolving IceScrum, a tool that supports the Scrum methodology. First, you select the source Eclipse project, or a piece of a project, that represents the existing system. IceScrum is a Java project, although MoDisco supports other technologies. Second, you analyze the source code and generate a KDM model, which is integrated into a KDM repository. Third, you choose KDM models and transform them into UML models representing the existing system. Fourth, you can apply refactoring patterns and introduce new functionalities to the UML models, at a higher abstraction level. You can make such modifications using the available MoDisco editors or other Eclipse plug-in tools. Finally, because MoDisco is an Eclipse plug-in, you can easily extend or integrate it with other tools—for example, code generators such as MOFScript, which can automatically generate source code for the same platform or a different target programming language.

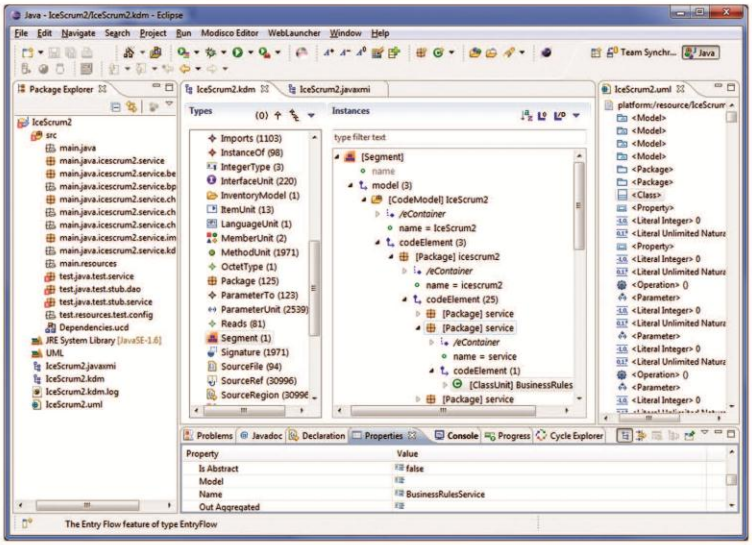


FIGURE A. Reengineering IceScrum with MoDisco. The interface shows source code, Knowledge Discovery Metamodel (KDM) representation, and Unified Modeling Language (UML) models for IceScrum.

Several support both traditional and model-driven reengineering.

Some commercial tools—for example, Microfocus Modernization Workbench, Amelio, BlueAge, and ObeoAgilit—focus on most of the reengineering stages available for I/O platforms and programming languages. Their costs can vary depending on the legacy system's platform, size, and other features. Others focus on one technique. For example, Resource Miner supports static analysis to obtain a code inventory and metrics, whereas CloneDR focuses on detecting and removing duplicate code.

Commercial integrated development environments, such as Visual Paradigm, are progressively incorporating more reverse-engineering techniques, such as automatic code generation from UML models (related to forward engineering) and reverse engineering from code to system design.

Reengineering research has also generated open source tools—for example, MOOSE, MoDisco, and Avalanche—that support ad hoc techniques proposed from academia. Some, such as MoDisco (see the sidebar), are also available as Eclipse plug-ins and can thus be combined with other related plug-ins.

The latest releases of some of the tools, such as Blue Age, Obeo Agility, MoDisco and MOOSE, already follow the model-driven approach.

Here are some hints for successful reengineering:

- Formalize the methodology, and use standards to carry out repeatable reengineering projects.
- Use well-known reengineering techniques and tools.
- Treat reengineering as a change management project with a concrete budget, skilled resources, and management support.
- Align the reengineering project with the company's strategic direction.
- Use techniques for decision support (for example, portfolio analysis) to focus on where reengineering has the biggest yield.
- Use good software engineering processes, such as for configuration control, documentation, and automatic verification.

The software industry has widely used reengineering. However, approximately 50 percent of reengineering projects fail because they produce unsatisfactory results or cost overruns.[5] This is owing to traditional reengineering, which is usually related to ad hoc solutions, and has two main limitations: standardization and automation. Standardizing and automating reengineering projects is necessary to better reuse legacy code and ensure that reengineered code can be further improved over time—independently of the chosen tools.

References

1. E.J. Chikofsky and J.H. Cross, "Reverse Engineering and Design Recovery: A Taxonomy," *IEEE Software*, vol. 7, no. 1, 1990, pp. 13–17.
2. G. Canfora, M.D. Penta, and L. Cerulo, "Achievements and Challenges in Software Reverse Engineering," *Comm. ACM*, vol. 54, no. 4, 2011, pp. 142–151.
3. C. Ebert and R. Dumke, *Software Measurement: Establish—Extract—Evaluate—Execute*, Springer, 2007.
4. *ISO/IEC DIS* 19506, *Information Technology—Architecture-Driven Modernization—Knowledge Discovery Meta-model (KDM)*, v1.1 (Architecture-Driven Modernization), ISO/IEC, 2009, p. 302; www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2=080&ics3=&csnumber=32625.
5. H.M. Sneed, "Estimating the Costs of a Reengineering Project," *Proc. 12th Working Conf. Reverse Eng.*, IEEE CS Press, 2005, pp. 111–119.

RICARDO PÉREZ-CASTILLO is a PhD student at the University of Castilla-La Mancha. Contact him at ricardo.pdelcastillo@uclm.es.

IGNACIO GARCÍA-RODRIGUEZ DE GUZMÁN is an assistant professor at the University of Castilla-La Mancha. Contact him at ignacio.grodriguez@uclm.es.

MARIO PIATTINI is a full professor at the University of Castilla-La Mancha. Contact him at mario.piattini@uclm.es.

CHRISTOF EBERT is the managing director of Vector Consulting Services. Contact him at christof.ebert@vector.com.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.

Call for Articles

IEEE Software seeks practical, readable articles that will appeal to experts and nonexperts alike. The magazine aims to deliver reliable information to software developers and managers to help them stay on top of rapid technology change. Submissions must be original and no more than 5,400 words, including 200 words for each table and figure.

Author guidelines: www.computer.org/software/author.htm
Further details: software@computer.org

www.computer.org/software

IEEE Software

CHAPTER 3

Research Proposal

> *MARBLE is a framework to recover business processes from legacy systems following the Architecture-Driven Modernization (ADM).*

3. RESEARCH PROPOSAL

This chapter provides a detailed description of the main contribution of this PhD Thesis. Firstly, Section 3.1 provides an overview of MARBLE, an ADM-based framework with which to retrieve business processes form legacy information systems. Secondly, Section 3.2 particularly provides a static technique framed in MARBLE. Finally, Section 3.3 provides a dynamic technique by considering runtime information also framed in MARBLE.

3.1. Overview

This section depicts the generic MARBLE framework. It provides a detailed explanation of the abstraction levels of MARBLE and the model transformations defined between these levels (see Figure 9). This framework can be considered as the guidelines needed to develop techniques with which to discover and retrieve business processes such as the static and dynamic techniques presented in Sections 3.2 and 3.3, respectively. This paper also introduces the business process archeology term as defined in this Thesis.

This section is presented through a journal paper published by the Information and Software Technology in 2011. This journal had an impact factor of 1.507 in 2010, according to Thomson's Science Citation Index (SCI). The reference of this paper is: *Pérez-Castillo, R., I.G.-R. deGuzmán, and M. Piattini, Business Process Archeology using MARBLE. Information and Software Technology, 2011. 53: p. 1023–1044.*

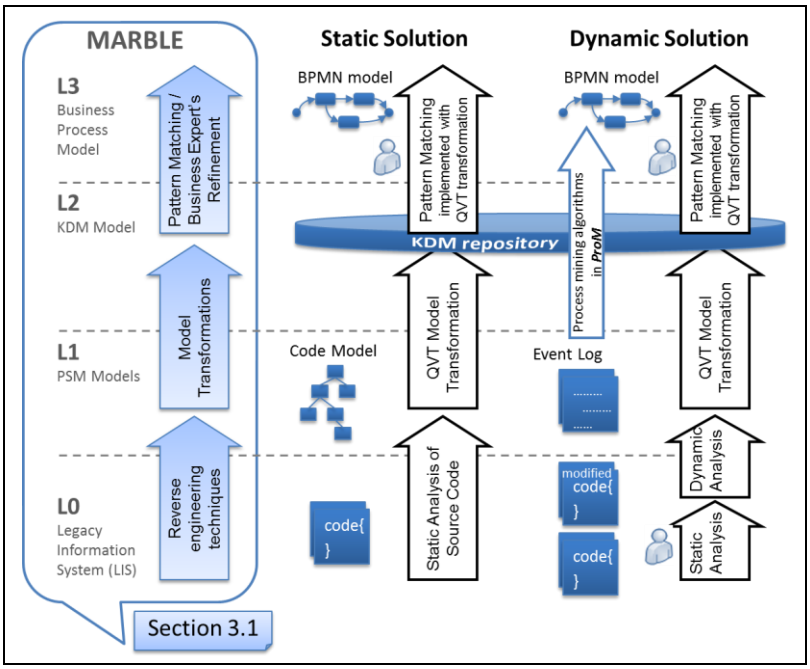


Figure 9. Main contribution of Section 3.1 as regards the proposal of the PhD Thesis as a whole

Information and Software Technology 53 (2011) 1023–1044

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Business process archeology using MARBLE

Ricardo Pérez-Castillo *, Ignacio García-Rodríguez de Guzmán, Mario Piattini

Alarcos Research Group, University of Castilla-La Mancha, Paseo de la Universidad 4, 13071 Ciudad Real, Spain

ARTICLE INFO

Article history:
Received 26 July 2010
Received in revised form 14 April 2011
Accepted 15 May 2011
Available online 24 May 2011

Keywords:
Business process archeology
ADM
Modernization
KDM
Legacy system
Case study

ABSTRACT

Context: Legacy information systems age over time. These systems cannot be thrown away because they store a significant amount of valuable business knowledge over time, and they cannot be entirely replaced at an acceptable cost. This circumstance is similar to that of the monuments of ancient civilizations, which have aged but still hold meaningful information about their civilizations. Evolutionary maintenance is the most suitable mechanism to deal with the software ageing problem since it preserves business knowledge. But first, recovering the underlying business knowledge in legacy systems is necessary in order to preserve this vital heritage.
Objective: This paper proposes and validates a method for recovering and rebuilding business processes from legacy information systems. This method, which can be considered a business process archeology, makes it possible to preserve the business knowledge in legacy information systems.
Method: The business process archeology method is framed in MARBLE, a generic framework based on Architecture-Driven Modernization (ADM), which uses the Knowledge Discovery Metamodel (KDM) standard. The proposed method is validated using a case study that involves a real-life legacy system. The case study is conducted following the case study protocol proposed by Brereton et al.
Results: The study reports that the proposed method makes it possible to obtain business process models from legacy systems with adequate levels of accuracy. In addition, the effectiveness of the proposed method is also validated positively.
Conclusion: The proposed method semi-automatically rebuilds the hidden business processes embedded in a legacy system. Therefore, the business process archeology method quickly allows business experts to have a meaningful understanding of the organization's business processes. This proposal is less time-consuming and more exhaustive (since it considers the embedded business knowledge) than a manual process redesign by experts from scratch. In addition, it helps maintainers to extract the business knowledge needed for the system to evolve.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Organizations have a vast number of highly functional and operational legacy information systems. These legacy systems are not immune to software ageing problems [49], and they must be maintained and evolve over time. System evolution can be motivated by new user requirements, technological or platform migration, and the most important reason: new business opportunities that must be addressed and supported by the organization's systems [29].

As a consequence of software evolution, legacy information systems embed meaningful business knowledge over time [28]. Thus, legacy systems become a key asset for organizations, since they can serve as a source to obtain information about the business processes governing the organization. Business processes define a sequence of activities in an organizational environment that realize a business goal [51]. If organizations know their business processes, they can manage them in order to maintain and improve their competitiveness level [21]. In addition, business knowledge is very important for organizations because it is usually unknown or incomplete, since a lot of business knowledge is embedded in legacy information systems that support the business' operation.

This paper presents a business process archeology method to rebuild business processes embedded (or buried, using the archaeology metaphor) in legacy information systems (ancient artifacts and relics). The proposed method allows business experts to explicitly state the business processes from the implicit business knowledge of their legacy systems. This proposal obtains business process models, which represents a good start point for business experts who can improve them (i) by adding manual activities that are not supported by information systems, or (ii) by adjusting the business processes to the real behavior of the organization. We think that the proposal performance is better than the performance

* Corresponding author. Tel.: +34 695662685; fax: +34 926295354.
E-mail addresses: ricardo.pdelcastillo@uclm.es (R. Pérez-Castillo), ignacio.grodriguez@uclm.es (I.García-Rodríguez de Guzmán), mario.piattini@uclm.es (M. Piattini).

0950-5849/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2011.05.006

of business process redesign by experts from scratch, because it does not discard the embedded business knowledge that is only present in legacy information systems, which usually is ignored by business experts.

The business process archeology method has been developed taking MARBLE as the starting point. MARBLE is a framework to recover business processes from legacy systems following the Architecture-Driven Modernization (ADM) approach [41]. ADM is the concept of modernizing legacy information systems with a focus on all aspects of the current system architecture and the ability to transform current architectures to target architectures [48]. ADM advocates carrying out reengineering processes following the principles of model-driven development, i.e., it treats all the software artifacts involved in the legacy system as models and can establish transformations between them. Therefore, MARBLE focuses on the reverse engineering stage of the reengineering process. MARBLE proposes four abstraction levels (with four different kinds of models) as well as three model transformations between them, in order to cover the whole path of the business process archeology method between legacy information systems and business processes. In addition, ADM defines the Knowledge Discovery Metamodel (KDM) standard [37], which presents a metamodel to represent and manage the legacy knowledge involved in all the different software artifacts of the legacy information systems. MARBLE uses this standard in order to represent the information recovered from legacy information systems as models.

MARBLE is a generic framework that specifies how to use the ADM and other related standards to recover business processes models from legacy information systems. Despite the fact that this proposal is aligned with MARBLE, presented in a previous work [41], this paper presents a more mature work which provides the specific analysis techniques and model transformations to achieve the first functional business process archeology method in order to be applied to real-life industrial projects. To date, the proposed method considers: (i) legacy source code as the key software artifact as the business knowledge source; (ii) the static analysis as the reverse engineering technique used to extract meaningful knowledge; (iii) a model transformation based on Query/Views/Transformations (QVT) to obtain a KDM model from the knowledge extracted by statically analyzing the source code; and (iv) other QVT model transformation supporting a set of business patterns to transform the KDM model into business process model.

Because the business process archeology method is framed in MARBLE, which uses ADM and KDM, the proposed method has three key advantages with respect to the other proposals in the literature:

- The formalization, repeatability and automation of the archeology method are possible due to the fact that the method follows the ADM approach. Thus, it reduces the costs of the maintenance activities that may use the recovered business process models. In turn, lower costs mean that the method improves the ROI (Return On Investment) on legacy information systems, thus also extending the lifespan of legacy information systems.
- Business knowledge management via the archeology method is carried out in an integrated and standardized manner with KDM.
- Since MARBLE follows model-driven principles that advocate that models can be interrelated through the different aforesaid abstraction levels, the business process archeology method improves feature location, i.e., identifying what elements of the business process were obtained from specific pieces of legacy source code. Feature location is an essential step in facilitating maintenance activity.

Additionally, this paper reports on a case study that applied the business process archeology method to an enterprise information system at a chemical laboratory company to recover its business processes. Firstly, a specific method to recover the business processes from the information systems based on Java was developed followings the MARBLE framework. Secondly, the case study was carried out using the protocol template for case studies proposed by Brereton el al. [3] in order to improve the rigor and validity of the case study. The results showed that the business process archeology method can obtain business processes in linear time with respect to the size of the legacy systems, and in addition, can obtain business processes that represent the business operation of the organization with reasonable levels of reliability.

The rest of this paper is organized as follows. Section 2 introduces the background of the paper: business process archeology and the ADM standard. Section 3 summarizes the work related to the business process recovery from the legacy information systems. Section 4 presents the proposed business process archeology method using MARBLE. Section 5 briefly shows the most important details of a tool developed to support the proposed method. Section 6 presents the planning and execution of the case study involving a real-life company system. Finally, Section 7 discusses conclusions and future work.

2. Background

This section presents the two main concepts used in this paper: business process archeology and the ADM approach and its standards.

2.1. Business process archeology

A business process is a set of coordinated activities and tasks performed in an organization, which aims at a specific business goal [51]. Interest inside organizations in knowing their business processes has increased because they consider them to be a key asset (i) for their performance, since business processes depict the organization's operations; and (ii) to improve their competitiveness, because business processes can be adapted to increase customer satisfaction, reduce costs, distinguish products or services, and so forth.

In order to achieve optimal business process management, it is necessary to represent business processes by means of a notation easily understood by the different agents involved in their management. The most important notations are UML 2.0 Activity Diagrams and Business Processes Model and Notation (BPMN) [38]. The Business Process Diagram (BPD) of BPMN is the notation used in this work since it is a well-known graphical notation and is easily understood by system analysts as well as business analysts.

Legacy information systems can be an obstacle to business process management, since these systems have evolved independently from the organization's business processes [15]. According to [40], *"a legacy information system is any information system that significantly resists modification and evolution to meet new and constantly changing business requirements"*. [16] go further to state that the *"code becomes legacy code just about as soon as it's written"*. The independent evolution from the business processes is due to the uncontrolled maintenance of legacy information systems. Indeed, according to the study provided by Koskinen et al. [25], changes in a business process is the third criterion (among a list of 49 software modernization decision criteria) to modify the information systems. Unfortunately, source code modifications usually do not have any effect on original business processes, which do not have the respective change. This fact means that legacy systems embed a significant amount of business knowledge that is not defined in

the original set of business processes. In this scenario, business process archeology is necessary to obtain the current set of business processes.

Business process archeology[1] is the engineering activity that studies the business processes in an organization by analysing the existing software artifacts in that organization. The objective is to discover the business forces that motivated the construction of the enterprise information systems. Real archeologists investigate several artifacts and situations, trying to understand what they are looking at, i.e., they must understand the cultural and civilizing forces that produced those artifacts. In the same way, a business process archeologist analyzes different legacy artifacts such as source code, databases and user interfaces and then tries to learn what the organization was thinking to understand why the organization developed the information system. Let us imagine two parallel examples in order to compare traditional archaeology (A) and business process archeology (BPA):

- The Roman civilization emerges. (BPA) An organization has a business process, which may be formalized or not; this does not matter.
- The Roman civilization produces monuments, writings, documents, etc. The monuments and other artifacts undergo maintenance and alterations over time. (BPA) Business processes are implemented in information (and manual) systems or by means of descriptions of business processes, which also undergo maintenance and alterations over time.
- The ancient Roman civilization disappears but the remains of the monuments, documents, etc. survive the test of time. (BPA) The description of the business processes disappear (or not).
- The archeologists study the monumental remains together with documentation from that era to rebuild and understand the Roman civilization. (BPA) The business process archeologists analyze software artifacts by means of reverse engineering and other techniques to reconstruct the business processes. Software artifacts can already be in use, however the current use can differ of the original use for which those were conceived (i.e. business process and legacy information systems are not aligned). In addition, actors who know business processes and legacy information systems could are not present in the organization at that moment (e.g. maintainers are seldom the same people who initially developed the information system).

2.2. Architecture-Driven Modernization (ADM)

The increasing cost of maintaining legacy systems, together with the need to preserve business knowledge, has turned the modernization of legacy systems into a significant research field [7]. ADM can be considered a mechanism for software evolution, i.e., it makes it possible to modernize legacy information systems and eradicates, or at least minimizes, the negative effects of the software ageing phenomenon in legacy systems. According to [34], ADM is the process of understanding and evolving existing software assets, and in addition, it restores the value of existing applications.

ADM solves the automation and formalization problem of traditional reengineering since it carries out reengineering processes taking *model-driven* principles into account. ADM is based on reengineering, but it considers different models as input and output artifacts of the process, thus solving the formalization problem found in traditional reengineering.

After the horseshoe reengineering model [22] was adapted to ADM it became known as the horseshoe modernization model (see Fig. 1). The model consists of three stages:

- **Reverse engineering** is represented by the left side of the horseshoe. It analyzes the legacy system in order to identify the components of the system and their interrelationships. In turn, the reverse engineering stage builds one or more representations of the legacy system at a higher level of abstraction.
- **Restructuring** is represented by the curve of the horseshoe since this stage takes the previous system's representation and transforms it into another one at the same abstraction level. This stage preserves the external behavior of the legacy system.
- **Forward engineering** is represented by the right side of the horseshoe because it generates physical implementations of the target system at a low abstraction level from the previously restructured representation of the system.

Moreover, the horseshoe modernization model considers three different kinds of models with respect to the abstraction level [31]:

- **Computation Independent Model (CIM)** is a business view of the system from a computation independent viewpoint at a high abstraction level. CIM models are sometimes called domain models.
- **Platform Independent Model (PIM)** is a view of a system from the platform independent viewpoint at an intermediate abstraction level. PIM models abstract all implementation details.
- **Platform Specific Model (PSM)** is a technological view of a system from the platform specific viewpoint at a low abstraction level. A PSM combines the specifications in the PIM with the details that specify how that system uses a particular type of platform or technology.

Transformations between the different kinds of models are formalized by means of the Query/Views/Transformations (QVT) standard proposed by the OMG [36]. The QVT specification consists of two distinct but related languages: (i) *QVT-Operational* language, which is procedural in nature, and (ii) *QVT-Relations*, a declarative language. QVT makes it possible to define deterministic transformations between models at the same abstraction level or at a different level. As a consequence, the model transformations can be automated in the horseshoe modernization model.

In addition, the ADM Task Force in the OMG has defined the KDM standard [37], which has been also recognized as the ISO 19506 standard [20]. The KDM standard defines a metamodel for modeling all the different legacy software artifacts involved in a legacy information system. The KDM metamodel provides a comprehensive high-level view of the behavior, structure and data of the legacy systems [20], but it does not represent procedural models of the systems such as Unified Modeling Language (UML). While UML can be used to generate new code in a *top-down* way, ADM-based processes involving KDM start from the legacy code and build a higher level model in a *bottom-up* way [32].

KDM is a metamodel divided into layers representing both physical and logical *software assets* of information systems at several abstraction levels [37]. KDM separates knowledge about legacy information systems into various orthogonal concerns that are well-known in software engineering as *architecture views*. KDM consists of four layers of abstraction, each one based on the previous layer (see Fig. 2) [20].

[1] Archeology is "the scientific excavation and study of ancient human material remains" [2] Archaeological Institute of America. *AIA Web Portal*. 2010 [1/28/2010]; Available from: http://www.archaeological.org/. Also, archeology is defined as "the scientific study of material remains (such as fossil relics, artifacts, and monuments) of past human life and activities" [30] Merriam-Webster, in *Dictionary and Thesaurus – Merriam-Webster Online*. 2010.

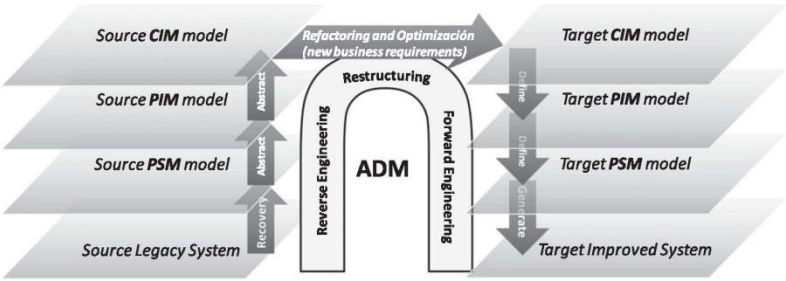


Fig. 1. Horseshoe modernization model.

- **Infrastructure Layer** is the layer at the lowest abstraction level and defines a small set of concepts used systematically throughout the entire KDM specification. There are three packages in this layer: Core, KDM and Source.
- **Program Elements Layer** offers a broad set of metamodel elements in order to provide a language-independent intermediate representation for various constructs defined by common programming languages. This layer represents implementation level program elements and their associations. This means that the program elements layer represents the logical view of a legacy system. This layer has two packages: Code and Action. Both packages define a single model, called the *CodeModel*.
- **Runtime Resource Layer** this enables the representation of knowledge about resources managed by the legacy system's operation environment, i.e., it focuses on those things that are not contained within the code itself. It has four packages: Data, Event, UI and Platform.
- **Abstraction Layer** defines a set of metamodel elements whose purpose is to represent domain-specific knowledge as well as provide a business-overview of legacy information systems. This layer has three packages: Conceptual, Structure and Build.

A very important challenge in business process archeology lies in the large conceptual gap that exists between business processes and legacy systems. This must be gradually reduced. ADM facilitates business process archeology by means of KDM, since it reduces the conceptual gap due to the fact that is organized into several layers at different abstraction levels [20]. Thus, KDM makes it possible to carry out endogenous transformations from models in lower layers to models in higher layers of the KDM structure. Therefore, specific business knowledge is extracted directly from the legacy system, at which point the implicit knowledge at a higher abstraction level can be inferred or deduced from the prior knowledge.

3. Related work

Business process archeology is a common and real problem that companies and academics have been trying to solve for many years. The static and dynamic approaches are the two main techniques to discover the current business processes. In addition, new advances in model-driven development field make it possible to improve the business knowledge extraction methods by means of model reuse, formalization and standardization.

Several works address business knowledge recovery from legacy information systems using the static analysis technique. Zou et al. [57] developed a framework based on a set of heuristic rules for extracting business processes following model-driven development principles. This framework statically analyzes the legacy source code and applies the rules to transform pieces of source code in business process elements. This work is based on the MDA approach, but it does not consider the ADM's standards as KDM. Beside source code, other software artifacts are also considered to obtain business processes in a static way, e.g. Ghose et al. [13] propose a set of text-based queries in documentation for extracting business knowledge, which is not based on the MDA approach. The intent of this approach for text-to-model extraction is to look for cues within text documents that suggest snippets of process models.

System databases are other artifacts used as input in static analysis, e.g. Paradauskas et al. [40] recover business knowledge through the inspection of the data stored in databases together with legacy application code. This work does not follow model-driven development principles. Another work taking database as the input artifact is provided by Perez-Castillo et al. [43], which proposes a reengineering framework to extract business logic from relational database schemas following the MDA approach.

Moreover, Wang et al. [50] present a framework for business rules extraction from large legacy information systems based on

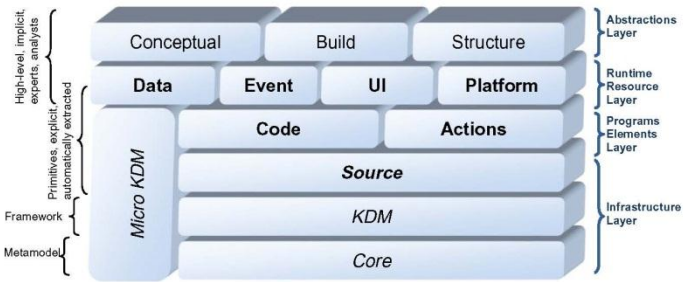


Fig. 2. Layers, packages, and concerns in KDM (adapted from [20]).

static program slicing. Program slicing is a reverse engineering technique consisting of the program decomposition into slices following a slicing criterion (e.g. fragments of source code that uses a specific program variable). However, the framework is not able to represent the recovered business knowledge in an understandable and standard way.

All these works solely rely on static analysis, which has the disadvantage that a lot of knowledge is lost since it disregards all runtime knowledge. Therefore, other solutions based on dynamic analysis have been suggested. According to the survey provided by Cornelissen et al. [6], dynamic analysis has been applied for a wide set of topics. For instance, Eisenbarth et al. [10] present a feature location technique based on dynamic analysis, which gathers the information from a set of scenarios invoking the features. These scenarios are previously defined by domain experts in a manual way, and the obtained result is a mapping between general and specific computational units with respect to a given set of features.

In order to extract complex business processes that are triggered by external actors, Cai et al. [4] propose an approach that combines the requirement reacquisition with a dynamic and static analysis technique. Firstly, the use cases are recovered through interviews with the users of the legacy information system. Secondly, according to those use cases, the system is dynamically traced. Finally, the traces obtained in run-time are statically analyzed to recover the business processes. Moreover, Di Francescomarino et al. [9] recover business processes by dynamically analyzing the Web application GUI-forms which are executed during user's navigation.

Other works addressing the dynamic approach provide process mining techniques that register event logs focusing on Process-Aware Information Systems (PAIS), i.e., process management systems such as Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) systems. The nature of these systems (in particular their process-awareness) facilitates the registration of events throughout process execution. Event logs depict the sequence of business process' activities executed, and can be used to discover the current business processes. In this sense, Günther et al. [14] provides a generic import framework for obtaining event logs from different kinds of process-aware information systems. In addition, Ingvaldsen et al. [18] focus on ERP systems to obtain event logs from the SAP's transaction data logs. Both works do not follow the model-driven development principles.

Dynamic analysis is more powerful than static analysis, since there is specific knowledge that is known only at run time. However, the dynamic analysis usually requires source code modifications in order to aggregate traces to register system execution information in an appropriate way. However, source code modifications are not always possible since legacy information systems can be in the production stage. Moreover, another point of controversy related to the usage of dynamic analysis techniques in realistic environments is that it might be thought that a company or organization would not accept the use of an automatically modified version of its information system. For this reason, business process archeology methods based on dynamic analysis technique should ensure that the source code modification is carried out without affecting the behavior of the original information system, i.e. in a non-invasive way.

This paper proposes a business process archeology method framed in MARBLE, which is based on the ADM approach together with the KDM standard. MARBLE is a generic model-driven framework that can use any reverse engineering technique to extract business knowledge, including static and dynamic analysis, in addition to other techniques. Furthermore, the nature of MARBLE vis-à-vis software artifacts is also hybrid, since MARBLE can extract business knowledge from any software artifact as well as business expert information. Specifically, the proposed business process archeology method uses static analysis as the main technique and focuses on legacy source code and manual intervention by business experts.

Table 1 presents a comparison of the different business process archeology methods proposed in the literature as well as the proposed business process archeology method. Each recovery process is framed in a matrix of software artifacts or knowledge sources in rows, along with the mechanism used to extract the business knowledge in columns.

4. Business process archeology method

The proposed method is framed in MARBLE framework, which specifies how to use ADM and other related standards like KDM

Table 1
Comparison between recovery processes.

Extraction Technique / SW artifact	Not Model-Driven		Model-Driven	
	Static Analysis	Dynamic Analysis	Static Analysis	Dynamic Analysis
External expert information	*Cai et al.* [4]	*Eisenbarth et al.* [10]	*** Proposed Business Process Archeology Method**	
Source code	*Wang et al.* [50]		*Zou et al.*[57]	
Documentation	*Ghose et al.* [13]			
Database	*Paradauskas et al.* [40]		*Pérez-Castillo et al.* [43]	MARBLE (Generic Framework)
User interfaces		*Di Francescomarino et al.* [9]		
Event Logs		*Günther et al.* [14] *Ingvaldsen et al.* [18]		

to recover business process from legacy information systems. The main purpose of MARBLE is the guidelines definition to achieve model-driven reengineering of legacy information system to recovery and represent meaningful knowledge at the top abstraction level (i.e., business process models). Thus, the business process models can be used to modernize legacy information systems preserving the meaningful business knowledge.

Therefore, MARBLE does not only meet the demands detected by Khusidman et al. [23], but also is aligned with the research agenda developed by the Software Engineering Institute (SEI) [26]. This report advocates using software modernization as a tool to obtain business processes from legacy information systems in order to migrate them to Service-Oriented Architecture (SOA) systems. In any case, MARBLE is a generic and extensible framework that can be used with different software modernization purposes.

MARBLE, the general-purpose and extensible ADM-based framework, is presented in Section 4.1. After that, the three specific transformations defined within our proposed business process archeology method are explained in detail in following subsections.

4.1. MARBLE

Modernization Approach for Recovering Business processes from LEgacy Systems (MARBLE) [41] is a framework that focuses on the reverse engineering stage in the horseshoe modernization model to obtain business processes from legacy information systems.

KDM is the core of MARBLE, since it enables the representation and management of knowledge extracted by means of reverse engineering from all the different software artifacts of the legacy system in an integrated way. Then, that legacy knowledge is gradually transformed into business processes. For this purpose, MARBLE is divided into four abstraction levels with three transformations among them (see Fig. 3).

The four generic abstraction levels proposed in MARBLE are the following:

- **Level L0**: This level represents the legacy information system in the real world, and is the source system to recover underlying business processes.

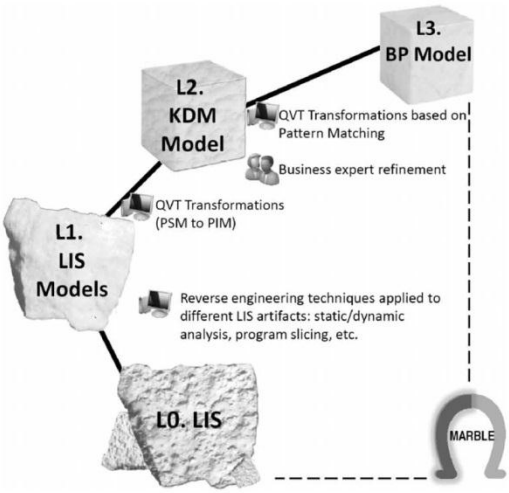


Fig. 3. MARBLE, the framework to support business process archeology.

- **Level L1**: This level represents several specific models, i.e., one model for each different software artifact involved in the archeology process like source code, database, user interface, and so on. These models are considered PSM models since they depict the software artifacts according to their specific technology platforms.
- **Level L2**: This level consists of a single PIM model that represents the integrated view of the set of PSM models in L1. The KDM metamodel is used for this purpose, since it makes it possible to model all the artifacts of the legacy system in an integrated and technological-independent manner. Firstly, we say that L2 is obtained in an integrated way because L2 works as a KDM repository that can be progressively populated with knowledge extracted from the different information systems of an organization. This is due to the fact that the *structure* package at the last KDM abstraction layer (c.f. Section 2.2) allows representing different systems, subsystems, components, architecture view, and so on. This is a key advantage because business processes are usually executed by multiple systems within an organization. Secondly, we say that L2 is represented in a technological-independent way due to the fact that KDM standard abstract (from the program element layer, the second layer of the KDM metamodel) all those details concerning the technological viewpoint (e.g. the program language). Thereby, the KDM repository at L2 can represent several legacy information systems even when their program languages or platforms are different.
- **Level L3**: This level depicts, at the end of the archeology process, the business processes recovered from the legacy system that is fully represented by a KDM model in L2. The business process model in L3 represents a CIM model of the system.

The three generic transformations between the four abstraction levels proposed in MARBLE are the following:

- **Transformation L0-to-L1**: This transformation obtains PSM models from each legacy software artifact. Classical reverse engineering techniques [5] such as static analysis, dynamic analysis, program slicing and dicing, formal concept analysis, subsystem decomposition, and so on, could be used to extract the knowledge from any software artifact and build the PSM model related to it. These PSM models are represented according to specific metamodels. For example, a Java metamodel may be used to model the legacy source code, or an SQL metamodel to represent the database schema, etc.
- **Transformation L1-to-L2**: The transformation between levels L1 and L2 consists of a set of model transformations to obtain a PIM model based on the KDM metamodel. This PIM model is built from the PSM models from level L1. The L1-to-L2 transformation can be implemented by means of QVT. The transformation from the legacy information system (L0) to the KDM model (L2) is not direct due to the fact that, in many cases, the platform-specific knowledge in the intermediate level L1 might be used to infer more business knowledge. Thus, the semantic gap between the legacy system and its KDM model is reduced incrementally through L1.
- **Transformation L2-to-L3**: This transformation is based on a set of patterns. When a specific structure is detected in the KDM model from level L2, each pattern indicates what elements should be built and how they are interrelated in the business process model in L3. This is known as pattern matching and can be implemented in MARBLE by means of QVT relations, the declarative part of the QVT language. In addition, this last transformation can be assisted by business experts who know the organization. The external information provided by experts also serves as valuable knowledge in business process archeol-

ogy. Experts can determine inconsistent or incoherent fragments in the preliminary business process models obtained after pattern matching; they can refactor the business process models, and incrementally fit the process models to the real behavior of the organization. Moreover, this transformation supports the representation of several subsystems at the same time. This transformation structures subsystems as different pools and lanes in the process model. Lanes and pools, which represent BPMN sub-modules, are related between them by means of certain data objects. These data objects are written by a sub-module and are read by another sub-module (e.g. data stored in a database by a subsystem and loaded by other subsystem; or a subsystem invoking a service of another one).

Moreover, not all parts of business processes are executed by legacy information systems, i.e., there are a lot of manual business activities. Manual business activities can be also added by final business expert intervention. Since we are proposing an automated approach, the treatment of that kind of activities is beyond of the scope of this work. Other works (e.g. do Nascimento et al. [10]) address challenges like identifying human activities in the legacy system, activities outside the legacy system, identifying roles, and so on.

The main objective of MARBLE is to provide a first version of business process models that, compared with business process redesign by business experts from scratch, represents a more efficient solution and a good start point to achieve business process archeology. In addition, the business process redesign by experts from scratch might discard meaningful business knowledge that is only present in legacy information systems. Anyway, MARBLE supports the addition and integration of manual business activities together the discovered business processes. For this purpose, MARBLE defines a finite set of operations that business experts can carry out in the business processes at the end of the automate part of our method.

MARBLE is defined as a generic framework to rebuild business processes because it defines the organization of the four abstraction levels as well as the guidelines to perform the three transformations between them. However, this generic framework must be instantiated for each kind of program language on which the system is based, for each reverse engineering technique that is considered to extract the legacy knowledge, and so forth. The following subsections depict the three specific transformations that show how the business process archeology is carried out.

4.2. L0-to-L1 transformation

In the proposed method, the L0-to-L1 transformation is characterized by the reverse engineering technique and the software artifact used. In this case, static analysis is the reverse engineering technique and the legacy source code is the chosen artifact. Static analysis consists of syntactically analyzing the source code of each source file that belongs to the legacy system.

The advantage of this reverse engineering technique is that a syntactic parser for analyzing the source code is easy and inexpensive to build. Moreover, the importance of the source code is underscored in legacy systems since important business knowledge is actually buried in the source code [33].

Static analysis has some limitations, for instance, this technique cannot extract information about the most used path of the source code in a specific source file when it is running. In this case, dynamic analysis, which consists of analyzing the source code execution, can complement the static analysis. However, dynamic analysis is usually difficult to implement because it requires modifying the source code by means of traces which are monitored throughout the execution. Modification of the source code of a legacy system in production is not an easy task. For this reason, static analysis was chosen as the reverse engineering technique for the proposed business process archeology method.

In addition, this transformation is specifically tuned to analyze Java-based systems. Therefore, while the static analysis is digging up the information from a Java source file, a source code model is built on the fly according to a Java metamodel (see Fig. 4). This means that there will be a source code model for each source code file at the end of the analysis. Each model is an abstract syntax tree of each source code file and these models represent the PSM models in the L1 level of MARBLE.

To illustrate the transformation, we present a running example for a small Java file, *Example.java*, which contains a *main* class with two additional methods (see the central panel of the MARBLE tool in Fig. 5). After executing the L0-to-L1 transformation, an abstract syntax tree is obtained (see the right panel in Fig. 5). In order to demonstrate how a method and its invocations are modeled, we present the respective node related to the main method in an ex-

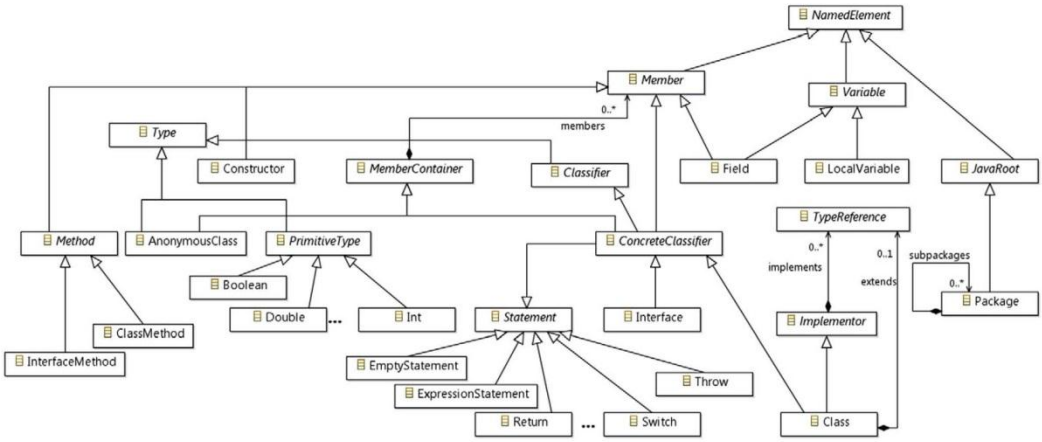


Fig. 4. A simplified view of the Java metamodel.

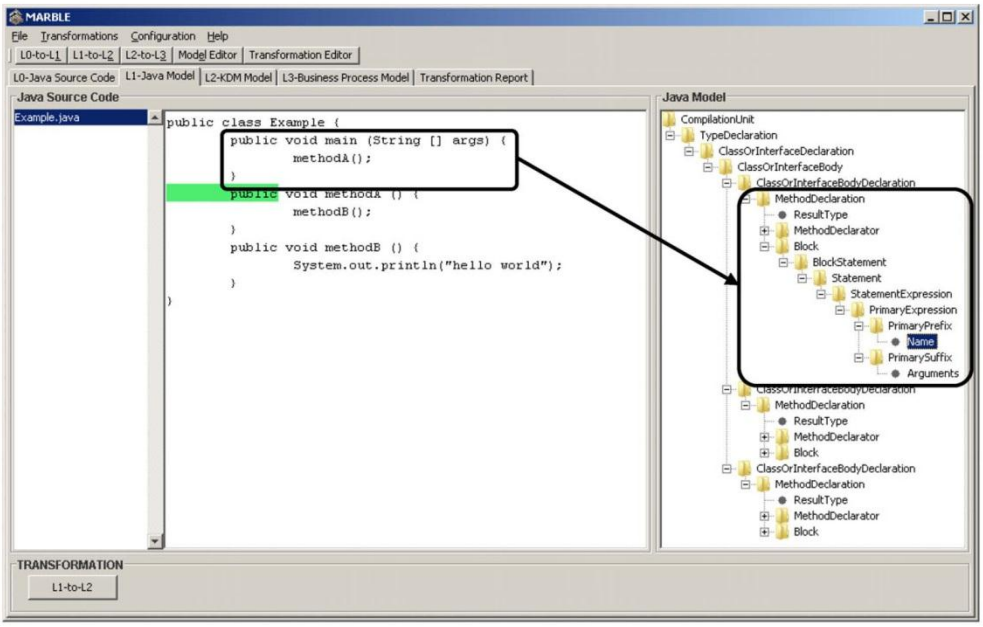


Fig. 5. A running example for a small Java file after the L0-to-L1 transformation.

panded way. The statement 'methodA();' is represented within the instance of *BlockStatement* metaclass as a *Statement* with a nested *StatementExpression*. In turn, the *StatementExpression*'s instance contains a *PrimaryExpression* with a *PrimaryPrefix* and a *PrimarySufix*. The instance of the *PrimaryPrefix* metaclass has a name representing the name of the method that is invocated (i.e., method); and the instance of *PrimarySufix* metaclass has an empty list of arguments (see the right panel of the tool in Fig. 5). This abstract syntax tree represents the code model at L1 level of MARBLE.

4.3. L1-to-L2 transformation

The L1-to-L2 transformation obtains a single PIM model according to the KDM metamodel. The KDM model integrates all the source code models obtained in the previous transformation.

Since the source code is the sole software artifact that is considered in the first transformation, this transformation only uses specific packages (portions of the metamodel) of the KDM to represent the source code: the packages *Code* and *Action* within the *Program*

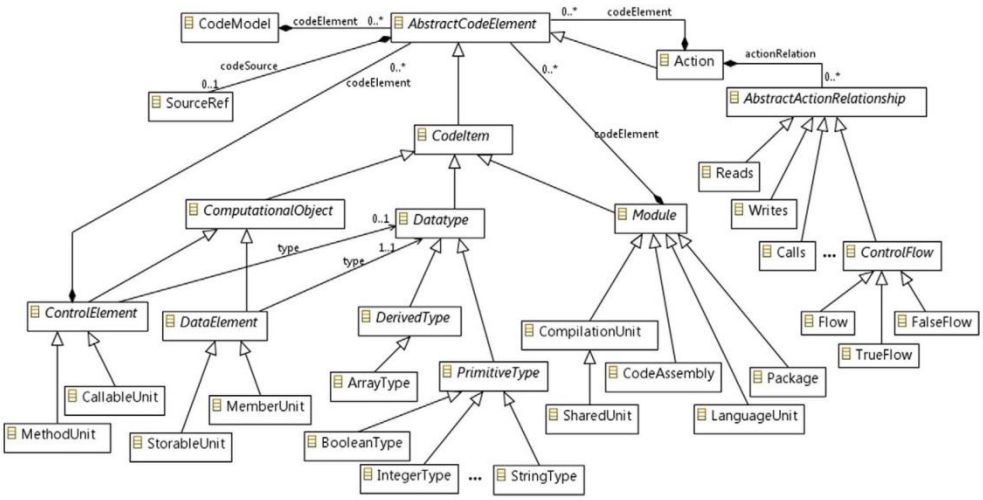


Fig. 6. A simplified view of the Code and Action packages of the KDM metamodel.

```
transformation Java2KDM (java:java, kdm:KDM){
   top relation package2package {
      packageName : String;
      checkonly domain java jp : java::containers::Package {
         name = packageName
      };
      enforce domain kdm kp : KDM::code::Package {
         name = packageName
      };
      where {
         jp.compilationUnits->forAll (jc:java::containers::CompilationUnit |
            jc.oclIsKindOf(java::classifiers::Class) implies class2compilationUnit
            (jc.oclAsType(Class), kp));
      }
   }
   relation class2compilationUnit {
      className : String;
      checkonly domain java jc : java::classifiers::Class {
         name = className
      };
      enforce domain kdm kp : KDM::code::Package {
         codeElement = kcu : KDM::code::CompilationUnit {
            name = className
         }
      };
      where {
         jc.members->forAll (jm:java::members::Method |
            jm.oclIsKindOf(java::members::Method) implies method2callableUnit
            (jm.oclAsType(Method), kcu));
      }
   }
   ...
}
```

Fig. 7. Two QVT relations of the L1-to-L2 transformation.

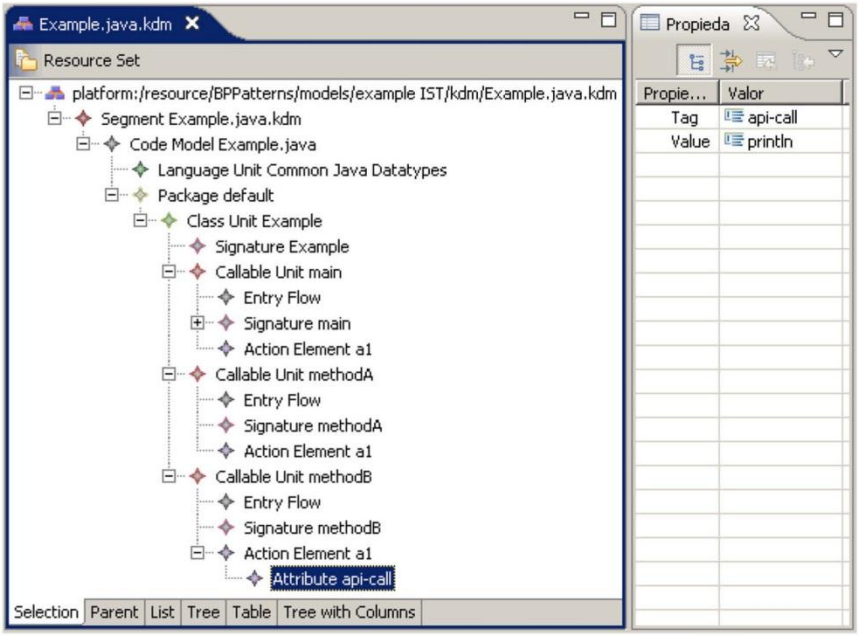


Fig. 8. An example KDM model obtained after the L1-to-L2 transformation.

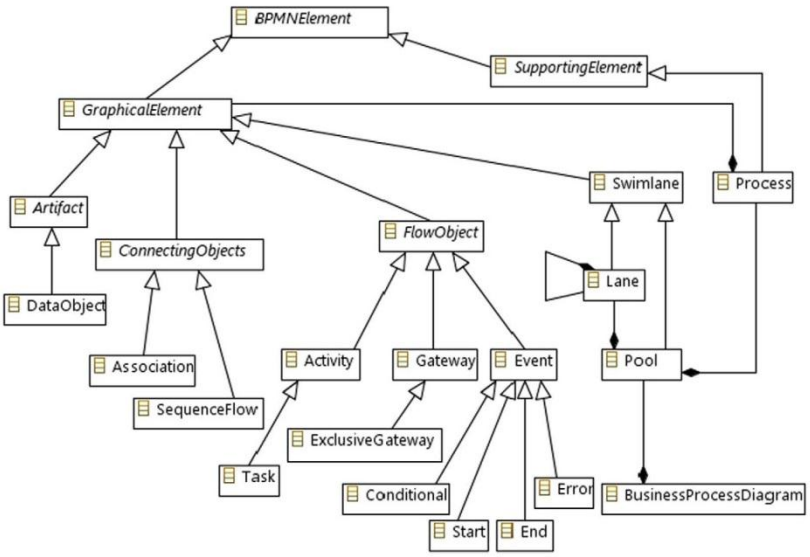


Fig. 9. BPMN metamodel.

Element layer in the KDM organization. *Program Elements* is the second abstraction layer of KDM after the *Infrastructure* layer and it aims to provide a language-independent intermediate representation for various constructs determined by common programming languages. There are two packages in this layer: *Code* and *Action*. The *Code* package represents the named items from the source code and several structural relationships between them and the *Action* package focuses on behavior descriptions and control- and data-flow relationships determined by them.

Fig. 6 shows the most important meta-elements of the *Code* and *Action* packages of the KDM metamodel. According to the KDM Code metamodel, each analyzed legacy system is represented as a *CodeModel* element, the root meta-element. A *CodeModel* is composed of *AbstractCodeElements*, a meta-element that represents an abstract parent class for all KDM entities that can be used as *CallableUnit*, *StorableUnit*, and so on. The *CodeElements* are also interrelated by means of *AbstractActionRelationships* (*Action* package), a meta-element representing an abstract parent for all KDM relationships that can be used to represent the code such as *Flow*, *Calls*, *Reads*, *Writes*.

The L1-to-L2 transformation is carried out by a model transformation, which is implemented through QVT *Relations*, the declarative part of the QVT language. The model transformation is almost direct in a declarative manner, since most of the transformations between the elements consist of renaming them according to the metamodels. This is because the structure of the Java metamodel and DM Code/Action metamodel are very similar.

For example, Fig. 7 shows two QVT relations involved in the L1-to-L2 transformation. The first relation named *package2package* is a top relation, i.e., it is always executed at the beginning of the QVT transformation. This relation transforms each Java package into a KDM package. The first relation calls to *class2compilationUnit* in the *where* clause for each Java class belonging to the Java package. The second relation transforms each Java class into a compilation unit in the output KDM model. In turn, this relation calls to the *method2callableUnit* relation for each method within the Java class, and so on.

To continue with the example introduced in previous section, we show the KDM model (see Fig. 8) obtained from the previous code model (see Fig. 5) by executing the L1-to-L2 transformation. The KDM model contains a *CodeModel* instance named '*Example*' which is nested within a *Segment* instance with the same name. The *CodeModel* instance has a package element with the name '*default*', since the original Java file does not define any package. The *Package* instance contains a nested *CompilationUnit* instance named '*Example*' as is defined in the name of the original Java file, since it is obtained from the *Class* instance of the Java code model at L1. The '*Example*' *CompilationUnit* instance contains an instance of the *CallableUnit* metaclass for each Java method at L1 (compare Figs. 5 and 8). Each *CallableUnit* instance is also defined by means of different *CodeElement* and *ActionElement* instances. For example, '*methodB*' is modeled as a *CallableUnit* containing (see Fig. 8): (i) an *EntryFlow* instance that defines the first KDM action in the unit (the first Java statement, since a model's sequentiality of actions is not clearly defined); (ii) a *Signature* instance that defines the parameters of the unit; and finally (iii) an *ActionElement* instance that represents the statement which calls to the external '*println*' function. The remaining *CallableUnit* instances follow a similar structure.

4.4. L2-to-L3 transformation

The L2-to-L3 transformation is the last transformation and obtains a business process model from the KDM model from a set of business patterns and a set of specific modifications by business experts.

The business process models in level L3 are represented according to the metamodel of BPMN (see Fig. 9). The BPMN metamodel represents *business process diagrams* (BPD), which involve four kinds of elements: (i) flow object elements such as *events*, *activities* and *gateways*; (ii) connecting object elements like *sequence flows*, *message flows* and *associations*; (iii) artifact elements such as *data objects*, *groups* and *annotations*; and (iv) swim lane elements for grouping elements such as *pools* and *lanes*.

The set of patterns in the L2-to-L3 transformation takes well-defined business patterns from the literature that have been successfully applied by business experts to model business processes [1,42,55,56]. Particularly, the set of patterns comes from the previous work presented in [42,44], although the implementation of these patterns has been substantially improved. For example, the name of the main methods is now transformed into a callable unit in the KDM model by using the name of the main class, since this name is more representative. On the other hand, some pre- and post-conditions have been also improved in their implementation.

Business patterns establish what specific structure of the legacy system (represented in a KDM model) is transformed to each specific business pattern. Therefore, each pattern consists of (i) a source configuration or structure of elements in the input model (KDM model in L2), and (ii) a target structure of elements in the output model (BPMN model in L3). Table 2 shows the set of patterns defined for this business process archeology method, which provide an assertion for each pattern in first-order predicate logic and the target graphical representation in business process models.

The set of patterns is defined in terms of KDM and BPMN elements, and thus the last transformation is independent of the program language and platform of the legacy system in contrast to the first and second transformations. This means that the set of proposed patterns could also be used with other systems in addition to Java-based systems.

The set of patterns is implemented by means of QVT relations, since the patterns can be easily implemented in a declarative manner. The QVT transformation consists of one or more QVT relations for each pattern. Each QVT relation defines an input domain (*check-only*) to depict the elements that are checked in the KDM model and an output domain (*enforce*) to represent the elements that are built according to the pattern.

Fig. 10 shows four QVT relations as an example using graphic notation. At the beginning, the relation *R1.CodeModel2BPD* is executed and it generates a business process diagram for each KDM model. The second relation *R2.Package2Pool* is invoked for each package from the *where* clause of *R1*, which in QVT can be defined by means of Object Constraint Language (OCL) [15]. The relation *R2* generates, in the business process diagram (the root element), a pool element that contains a business process, both with the same name as the package according to pattern *P1. BPD skeleton*.

Each source code package is considered as the minimal unit to be transformed into a business process model. This assumption will provide processes with lower accuracy levels than solutions considering correlation data set at the beginning of the process. However, the advantage of this solution is that it does not require the initial correlation information, and it automatically provides a preliminary business processes by statically analyzing the legacy source code. Those business processes can be used as the basis to understand the entire business processes of the organization, which can be refined by business expert at the end.

At the end of the R2 execution, the *where* clause in R2 invokes the relation *R3.MethodUnit2Task*, which creates the tasks related to the method unit elements according to the pattern *P2. Sequence*. This relation transforms methods into tasks, and invocations between them into sequence flows since our proposal follows the well-known *"a callable unit-a-task"* approach proposed by Zou et al. [57]. There are other choices that transform methods and their respective calls into subprocess-task relationships (i.e. composite tasks). However, the *"a callable unit-a-task"* approach is better to solve problems regarding the method granularity, since it can treat all the methods in the same manner and then, it can discard fine-grained methods using heuristics like removing the 'getters/setters' methods.

Moreover, the R3 relation invokes the relation *R5.WritesStorableUnit2DataObject*. The R5 relation implements the pattern *P6. Data Output*, thus generating, in the business process diagram, data objects associated with the tasks obtained previously by the relation *R3*.

The set of proposed patterns makes it possible to obtain preliminary business process models, but in order to successfully conclude this transformation, business experts can modify and improve the business process model after the pattern matching stage. The final manual intervention may be used by business experts to discover sub-modules by joining and splitting some preliminary processes into different pools. For example, data objects common in two different sub-modules could be transformed into message flows between those pools that represent sub-modules.

The L2-to-L3 transformation of the business process archeology method defines therefore a specific set of manual interventions that the business expert can carry out. Table 3 shows the finite set of five operations that are available for business experts to modify the business process models. Table 3 shows the description and elements in which the operation can be applied for each operation, since not all the operations can be used for all kinds of business process elements.

To conclude the running example, Fig. 11 shows the respective BPMN model obtained from the KDM model (see Fig. 8) by executing the L2-to-L3 transformation. The BPMN model contains a sequence of three tasks: *Example*, *methodA* and *methodB*, which are obtained from the three *CallableUnit* instances of the KDM model through the pattern *'P2. Sequence'* (compare Figs. 8 and 11). The first task is named *'Example'* instead of *'main'*, since the *P2* pattern changes the name of main methods for the name of its respective class in order to have a more representative name. Moreover, the invocation to the *'println'* function within *'methodB'* is transformed, according to the pattern *'P4. Collaboration'*, into a task with the same name and a round-trip sequence flow from the task *'methodB'*.

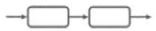
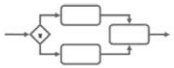
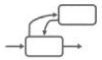
5. A tool to support business process archeology

A tool based on the *Eclipse* platform was especially developed to automate the proposed business process archeology method (see Fig. 12). The developed tool allows business process archeologists to complete the entire method, since it automates the three proposed model transformations. In addition, the tool aids the final manual intervention by business experts through a graphical editor of business process.

5.1. Technologies involved

This tool was developed for Java-based legacy systems and can be used to carry out case studies involving Web applications implemented in Java. The tool is based on four key technologies. The first technology is *JavaCC*, which is a parser and scanner generator for Java [39]. It is used to develop a static analyzer in the first transformation. The second technology is Eclipse Modeling Framework (EMF), which is a modeling framework and code generation facility for building tools and other applications based on structured data models [11]. This framework makes it possible to build specific metamodels according to the *ECORE* meta-metamodel (i.e. ECORE is the metamodel proposed by the Eclipse platform to define metamodels). Then, from these metamodels, EMF provides tools to produce a set of Java classes for the model, along with a set of adapter classes that enable viewing and command-based editing of the model as well as a basic editor. Another Eclipse framework, such as Graphical Modeling Framework (GMF), is also used together with EMF to generate graphical editors from the *ECORE* metamodels. Finally, the fourth technology is XML Metadata Interchange (XMI), which is a model-driven XML integration

Table 2
Patterns to obtain business process elements in L3 from legacy system elements in L2.

Pattern	Elements detected in L2	Structure in L3
P1. BPD Skeleton. This pattern creates the root structure of the BP model. It creates a BP diagram for each KDM code model. Also, it builds a pool element with a nested process in the BP diagram for each package of the KDM code model	$\forall x, y CODE_MODEL(x) \wedge COMPILATION_UNIT(y) \wedge COMPILATION_UNIT(y) \in CODE_MODEL(x) \Rightarrow BPD(x) \wedge POOL(y) \wedge PROCESS(y) \wedge PROCESS(y) \in POOL(y) \in BPD(x)$	
P2. Sequence. This pattern takes any callable piece of code from the KDM code model and maps them into tasks in the BP diagram. In addition, the sequence of calls to callable units is transformed into a set of sequence flows in the same order between the tasks built from the callable unit respectively	$\forall x, \exists y CALLABLE_UNIT(x) \wedge CALLABLE_UNIT(y) \wedge CALL(x, y) \Rightarrow TASK(x) \wedge TASK(y) \wedge SEQUENCE_FLOW(x, y)$	
P3. Branching. This pattern transforms each conditional jump of the source code that has two mutually exclusive choices into an exclusive gateway and two different sequence flows in the BP model. Typically those exclusive conditional branches are related to the *if ... then ... else* or *switch* clauses in several programming languages. The exclusive gateway represents the condition that is evaluated and the two sequence flows represent two conditional transitions that depend on the value (true or false) of the evaluation	$\forall x, \exists y, z CALLABLE_UNIT(x) \wedge CALLABLE_UNIT(y) \wedge CALLABLE_UNIT(z) \wedge TRUE_FLOW(x, y) \wedge FALSE_FLOW(x, z) \Rightarrow EXCLUSIVE_GATEWAY(x) \wedge TASK(y) \wedge TASK(z) \wedge SEQUENCE_FLOW(x, y) \wedge SEQUENCE_FLOW(x, z)$	
P4. Collaboration. Each call to external callable unit (i.e. API libraries or external components outside the legacy system) is transformed into an auxiliary task as well as two sequence flows: the first from the source task to the auxiliary task and the second returning to the source task	$\forall x, \exists y CALLABLE_UNIT(x) \wedge FOREIGN_CALL(x, y) \Rightarrow TASK(y) \wedge SEQUENCE_FLOW(x, y) \wedge SEQUENCE_FLOW(y, x)$	
P5. Data Input. This pattern builds a data object in the BP model for each input data within a callable unit in the KDM code model. Also, this pattern builds an association between the data objects and the task previously built from the callable unit. This pattern only considers as input data the parameters or arguments of the callable unit, but it does not consider the auxiliary variables within the callable unit	$\forall x, \exists y CALLABLE_UNIT(x) \wedge STORABLE_UNIT(y) \wedge READ(x, y) \Rightarrow TASK(x) \wedge DATA_OBJECT(y) \wedge ASSOCIATION(y, x)$	
P6. Data Output. Each piece of output data involved in a callable unit is transformed by means of this pattern into a data object as well as an association from the task (built from the callable unit) to the data object. This pattern excludes as output data the auxiliary and intermediate data in the body of the callable unit. The output data is the data returned by the callable unit or external data related to databases or files	$\forall x, \exists y CALLABLE_UNIT(x) \wedge STORABLE_UNIT(y) \wedge WRITE(x, y) \Rightarrow TASK(x) \wedge DATA_OBJECT(y) \wedge ASSOCIATION(x, y)$	
P7. Start. The task building from the callable unit that starts the execution of any program or application of the legacy system is considered the initial task. Therefore, a start event is built into the BP diagram and a sequence flow from this event to the initial task is also created	$\forall x, y TOPCALLABLE_UNIT(x) \Rightarrow START_EVENT(y) \wedge TASK(x) \wedge SEQUENCE_FLOW(y, x)$	
P8. Implicit Termination. This pattern builds an end event in the BP model. Then, it creates sequence flows from 'end task' and those flows merge in the end event. A task is considered an 'end task' if this task does not have any outgoing sequence flow	$\forall x, \nexists y, \exists e TASK(x) \wedge SEQUENCE_FLOW(x, y) \wedge END_EVENT(y) \Rightarrow SEQUENCE_FLOW(x, e) \wedge END_EVENT(e)$	
P9. Conditional Sequence. This pattern transforms each conditional call into a sequence flow fired under a conditional intermediate event through to the task related to the callable unit. This pattern makes it possible to create arbitrary cycles in the BP diagram	$\forall x, \exists y, i, \nexists z CALLABLE_UNIT(x) \wedge CALLABLE_UNIT(y) \wedge CALLABLE_UNIT(z) \wedge TRUE_FLOW(x, y) \wedge FALSE_FLOW(x, z) \Rightarrow TASK(x) \wedge TASK(y) \wedge CONDITIONAL_EVENT(i) \wedge SEQUENCES_FLOW(x, i) \wedge SEQUENCE_FLOW(i, z)$	
P10. Exception. Each call to callable unit under any exception is transformed into a task for the piece of source code that handles the exception as well as a sequence flow fired under an error intermediate event. Indeed, this pattern can be understood as a specialization of the previous pattern P9	$\forall x, \exists y, e CALLABLE_UNIT(x) \wedge CALLABLE_UNIT(y) \wedge EXCEPTION_FLOW(x, y) \Rightarrow TASK(x) \wedge TASK(y) \wedge ERROR_EVENT(e) \wedge SEQUENCES_FLOW(x, e) \wedge SEQUENCE_FLOW(e, z)$	

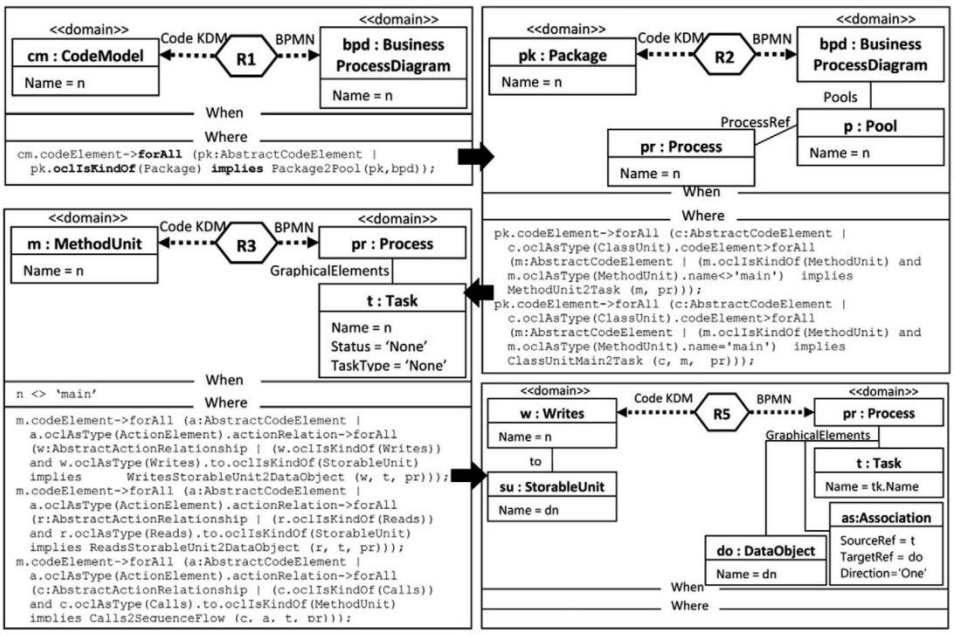


Fig. 10. Four QVT relations (graphical representation) of the L2-to-L3 transformation.

Table 3
Operations available to carry out manual intervention by business experts.

Operation	Description	Elements which it can be applied
Add [A]	This operation modifies the original business process model appending new elements to that model. For instance, business experts can use this operation to add manual activities that cannot be recovered from the legacy system	Activity, sequence flow, data object, association, gateways and event
Remove [R]	This operation modifies the original business process model deleting any element of that model. For instance, business experts can use this operation to drop activities recovered from auxiliary methods that are not related to the system's business domain	Activity, sequence flow, data object, association, gateways, event and business process model
Rename [RN]	This operation is used by business experts when they want to give another name to any element of the business process model, thus the new name accurately represents the semantics of the element	All the named elements
Join [J]	This operation makes it possible to consider two or more elements as only one element. Actually, it can be considered a transaction that consists of several *remove* and *rename* operations	Business process model and activity
Split [S]	This operation takes an element of the business process model and divides it into two or more elements. Actually, it can be considered a transaction that involves some *add* and *rename* operations	Business process model and activity

framework for defining, manipulating and interchanging XML data and objects [35]. Any model involved in MARBLE can be made persistent by means of an XMI file.

5.2. Tool modules

The tool is divided into three different modules that are aligned with each transformation in the proposed business process archeology method.

To support the first transformation, a tool module to carry out static analysis was developed. In this case, the module was built specifically for parsing Java source code. This tool module was developed through *JavaCC* from the *Extended Backus–Naur Form*

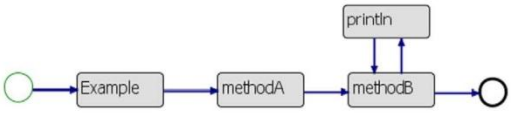


Fig. 11. An example BPMN model obtained after the L2-to-L3 transformation.

(EBNF) grammar of Java 1.5. This module takes a Java file as input and then generates an XMI file as the output that represents the Java code model, a PSM model in L1 (see Fig. 12A).

The second module executes a set of QVT transformations to obtain a KDM model in L2 from the Java code model obtained previously. The transformation is executed using the open source *Medini*

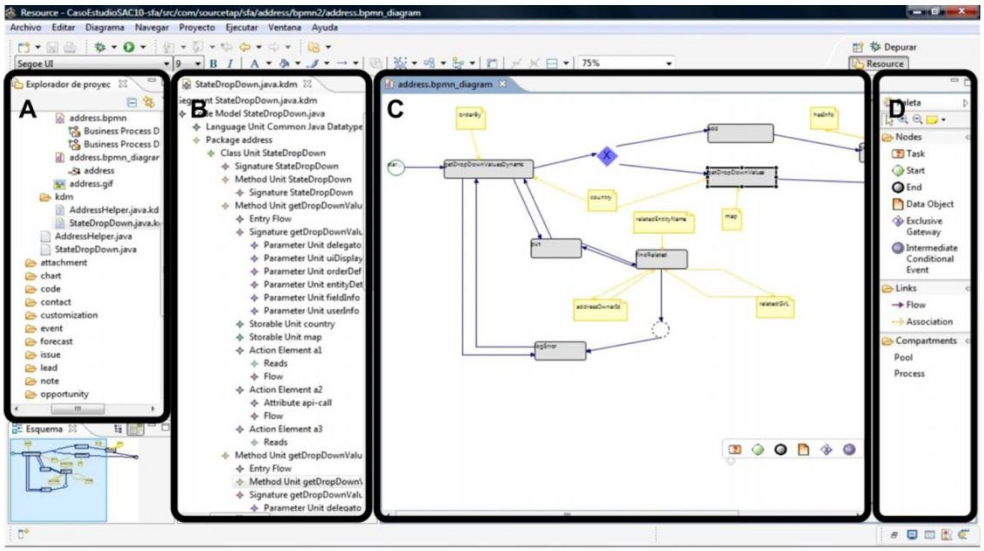


Fig. 12. The tool to support the business process archeology method.

QVT framework [17], a model transformation engine for *QVT Relations*. Also, it has a tree view editor for KDM models (see Fig. 12B) that was built through EMF.

The third module also executes a QVT transformation to support the third transformation based on pattern matching. Moreover, a graphical editor for business process diagrams in L3 was developed using EMF/GMF (see Fig. 12C). Therefore, this editor allows business experts to modify the preliminary business process models according to the proposed set of manual interventions (see Fig. 12D).

6. Case study

This section presents a detailed case study of a real-life legacy system where the underlying business processes are rebuilt following the proposed business process archeology method. The case study was conducted according to the methodology for designing, conducting and reporting case studies proposed by Brereton et al. [3]. In addition, the case study design was evaluated by means of the checklists for researchers proposed by Runeson et al. [47]. The use of a case study methodology improves the quality of the resulting case study, ensuring the rigorousness and repeatability of the study. The following subsections show the case study details according to the items proposed in the case study protocol template: background, design, case selection, case study procedure, data collection, analysis and validity evaluation.

6.1. Background

Firstly, the previous research on the topic must be identified. The related work presented in Section 3 shows other recovery processes to obtain business processes from legacy information systems, and it compares all the different recovery processes in the literature together with our proposal.

The works in the literature about business processes recovery are mainly characterized by three factors: (i) whether they follow the model-driven approach or not; (ii) the technique to obtain business processes, mainly static or dynamic analysis; and finally (iii) the legacy software artifacts considered to extract business knowledge.

This case study validates a business process archeology method framed in MARBLE. The proposed method is a model-driven framework for recovering business processes that uses static analysis as the recovery technique and source code as the legacy software artifact to extract business information. In addition, the method considers business expert knowledge as a complimentary source of business knowledge at the end of the process depicted in the method.

The object of study is the proposed method, and the purpose of this study is the evaluation of specific properties of the method related to effectiveness and efficiency. Thus, the main research question addressed by this case study is *MQ: Can the proposed method properly obtain business processes from legacy systems?* In addition, Table 4 shows the additional research questions that are identified from the *MQ* question. The additional research question *AQ1* is established in order to evaluate whether the business process models obtained from the legacy systems faithfully represent the business operation of the organization. This question is more related to the effectiveness of the proposed method. Moreover, the additional research question *AQ2* is proposed to evaluate whether the proposed business process archeology method obtains business processes efficiently. Both additional research questions make it possible to answer the main research question, *MQ*.

6.2. Design

This case study consists of a single case, i.e., it focuses on a single legacy information system to recover and rebuild its underlying business processes.

Table 4
Case study research questions.

Id	Research Question
MQ	Can the proposed method properly obtain business processes from legacy information systems?
AQ1	Can the proposed method obtain business processes with adequate accuracy levels?
AQ2	Is the proposed business process archeology method efficient?

The case study was designed according to the embedded design proposed by Yin [54], since the case study considers multiple units of analysis to be studied within the case. The analysis units are the different source code packages of the legacy information system. This is the independent variable of the study. Each package is transformed into a business process model according to the pattern matching supported through the L2-to-L3 transformation of the proposed method. Despite the fact that packages form the unit of analysis and a business process model is built for each package, the business process models can be joined or split after the manual intervention by business experts.

The objective of the study is to analyze the business processes in order to answer the questions established in Table 4. For this purpose, a set of measures was established to quantitatively answer the research questions for each business process model.

The study proposes two measures, *Precision* and *Recall*, in order to evaluate the effectiveness of the proposed method through the *AQ1* question. Both measures are used in retrieval information scenarios [45]. The *Precision* measure indicates the amount of relevant recovered elements within the set of recovered elements in a business process model. An element is considered relevant if it faithfully represents a business element in the real world. The *Recall* measure represents the fraction of relevant recovered elements of the total of relevant (recovered and not recovered) elements. These two measures are considered as dependent variables of the study.

$$PRECISION = \frac{|\{relevant\ tasks\} \cap \{recovered\ tasks\}|}{|\{recovered\ tasks\}|} \quad (1)$$

$$RECALL = \frac{|\{relevant\ tasks\} \cap \{recovered\ tasks\}|}{|\{relevant\ tasks\}|} \quad (2)$$

In order to apply these measures in a business process archeology scenario, we considered the task element as the unit element to be counted. Therefore, the *Precision* measure (1) is defined as the number of true relevant tasks divided by the total number recovered tasks, i.e., the sum of true relevant tasks and false relevant tasks that were incorrectly recovered. Moreover, the *Recall* measure (2) is defined as the number of true relevant tasks divided by the total number of relevant tasks, i.e., the relevant tasks of the business process and others tasks that should have been recovered but were not recovered.

Relevant tasks are those tasks in the model that the experts have in mind according to the organization's current business processes, i.e. the actual business process model used in the preliminary business process improvement. Business experts made decision about whether a particular recovered task is a relevant one. Thus, business experts check four conditions that should be met by relevant task. The first condition specifies that the task must represent a real-life business operation within the organization. This condition is not evaluated by considering task names, since these names are inherited from legacy code and they may be biased regarding the real business activity names provided by business experts. The second condition ensures that all the relevant tasks preceding the evaluated task must be recovered before the task under evaluation. In order to fulfill this condition, the predecessor tasks can be directly or indirectly connected to the task under evaluation, i.e., there could be non-relevant tasks intercalated between the evaluated task and their predecessor relevant tasks. In a similar manner, the third condition ensures that all the subsequent tasks must be directly (or indirectly) recovered relevant tasks. Finally, the fourth condition specifies that all the data objects related to the task under evaluation have been also recovered.

To evaluate the measures with the preliminary business process models, business experts from the organization must be selected. This study considers two people as business experts: the chief information officer (CIO) and chief operation officer (COO). The profile of these people involves knowledge about the organization's business processes: CIO knows the provided services, managed information and information technologies used in the organization; and COO is responsible for the daily operation of the organization. After obtaining preliminary business process models, the business experts scored the base measures like recovered relevant tasks, recovered non-relevant tasks, non-recovered relevant tasks, and so on. With this information, we calculate the derived measures, precision and recall. In order to merge the information provided by both business experts, we use the *Delphi* technique [46], which allows achieving a workable consensus within time limits.

Moreover, the answer to question *AQ2* is related to the time spent on executing the transformation. This time is automatically measured by the tool developed for each transformation between MARBLE levels. This measure is analyzed with respect to the total number of elements built into each specific business process model (for each analysis unit). Both the number of elements and the transformation time are also considered as dependent variables in this study.

6.3. Case selection

After designing the case study, the case under study must be selected. In order to select the most appropriate case, i.e., the most suitable legacy system, the criteria for the case selection are defined in Table 5.

Criterion C1 ensures that the selected legacy system is an information system that performs business operation management. For instance, this criterion discards embedded systems and real-time systems, which do not support the business operation of an organization or company. Also, criterion C2 ensures that the legacy system is a real-life system that is used in a production environment. Criterion C3 ensures that the selected system is really a legacy information system, i.e., the system has been maintained and modified from its original state. To evaluate this criterion, it uses the amount of modifications in the system that alter the business processes, i.e. the system modifications related to the *adaptive* and *perfective* maintenance according to ISO/IEC 14764:2006 [19]. The amount of changes is better than, for instance, the time in production stage, since that time does not imply that the system has been modified significantly. Criterion C4 guarantees that the legacy system is not a 'toy program', thus this criterion defines a threshold of 20,000 lines of source code. Finally, criterion C5 ensures that the system was built using Java language and therefore, the proposed process can be applied to the system.

After evaluating more than ten available systems using the above criteria, one company's legacy information system was selected for study. The legacy system is named *VillasanteLaboratory* and it manages the operation of a Spanish company in the water and waste industry. *VillasanteLaboratory* manages information related to chemical laboratories, customers and products such as chemical analysis, dilutions, and chemical calibrations. The analyses supported by the application examined different parameters, including a large number of physical, chemical and microbiological

Table 5
Criteria for case selection.

Id	Criterion for case selection
C1	It must be an enterprise system
C2	It must be a real-life system
C3	It must a legacy system
C4	It must be no smaller than 20 KLOC
C5	It must be a Java-based system

parameters according to current regulations and laws for controlling water quality. In addition, the system makes it possible to manage users and roles, meaning that it also defines a system administration module. As a consequence, *VillasanteLaboratory* meets C1 and C2. This legacy system has been in the production stage for four years, and it has had three major modifications with seven medium modifications in total. The version history (without minor modifications) was: 1.0; 1.1; 2.0; 2.1; 2.2; 3.0; 3.1; 3.2; 3.3; 3.4; 4.0. Thereby, the selected system also meets C3. From a technological point of view, *VillasanteLaboratory* is a traditional Web application and its architecture is separated into three layers: presentation, business and persistence. The technology used to develop the presentation layer was *JSP* (*Java Server Pages*), *Java* for the business layer and *SQL Server* together with *JDBD-ODBC* for the persistence layer. The total size of the legacy system is 28.8 KLOC. Criteria C4 and C5 are therefore satisfied with this system.

6.4. Case study procedure

After the case study design and the case selection, the case study was carried out. For this purpose, the case study procedure was established using the followings steps:

1. A set of meetings between the chief information officer, other company staff and the researchers must be carried out. As a result of these meetings, the legacy source code is given under informed consent and a non-disclosure agreement is signed to protect its business information. The business expert needed to perform the manual post verification to evaluate the measures is also selected in this step.
2. The legacy system under study is implanted. The source code is deployed in a Web server, the database schema is built by means of the given database scripts, and the initial data is loaded.
3. Once the system is operative in the experimental environment, the set of JSP pages is systematically scanned in the Web explorer to generate the whole set of associated servlet classes. Thus, all legacy source code is Java code and can be analyzed using the proposed tool.
4. The different business process models are obtained from the legacy source code using the tool developed to support the proposed business process archeology method. Fig. 13 shows an example of one of the business process models obtained to manage the customer information in the company.
5. The preliminary business processes obtained from the model transformation are analyzed by business experts. Business experts mark which task elements in the obtained business processes have been correctly recovered and which have not. In addition, business experts identify the tasks that should have been recovered but were not recovered. Therefore, the business expert intervention is used to evaluate the *Precision* and *Recall* measures.
6. The key information related to the generation of business processes, as well as the business expert intervention, is collected using the proposed business process archeology method. Section 6.5 presents the data collection plan defined for the case study.
7. The data collected in the previous stages is analyzed to draw conclusions to answer the research questions. Section 6.6 shows the results obtained from this case study.
8. Finally, the case study is reported and feedback is obtained from the company.

6.5. Data collection

The data to be collected and the data sources must be defined before starting the execution of the business process archeology method. In this case study, using the previous assumptions, the following data was recorded for each analysis unit (each source code package):

- Number of source code files representing the L0 level within each source code package.
- Number of business process models obtained after the L2-to-L3 transformation. This transformation always builds a single model for each unit analysis.
- The total number of elements in the business process model, and especially, the total number of task elements, since this is the unit to evaluate the *Precision* and *Recall* measures.
- The transformation time in milliseconds are given by the tool and are also annotated to answer question *AQ2*.

Table 6 shows the data obtained for each source code package after the execution of the transformations. The table shows all

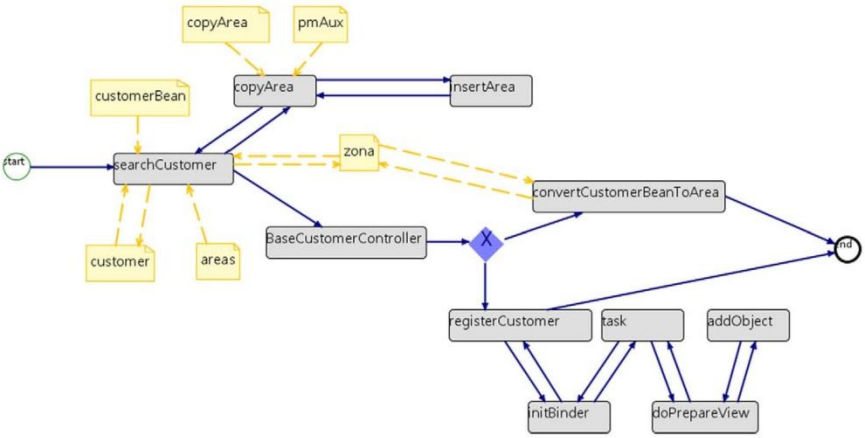


Fig. 13. An example of a BPMN model obtained in the case study.

Table 6
Data collected in the case study.

Package	# Source code files	# BPMN models	# Elements	# Tasks	Transf. time (ms)	Preliminary BP Id	Manual intervention in business process models
src	2	1	44	8	646	1	R
security.manager	2	1	24	3	222	2	J[3,21], RN["Security Management"]
security.utils	3	1	12	2	153	3	J[2,21], RN["Security Management"]
dao	40	1	370	191	1243	4	R
manager	36	1	195	204	791	5	R
model	21	1	46	42	471	6	J[7], RN["Administration"]
web	2	1	18	3	159	7	J[6], RN["Administration"]
web.analysis	23	1	419	80	3974	8	RN["Chemical Analysis Management"]
web.calibrations	11	1	177	39	1184	9	RN["Chemical Calibration Management"]
web.customer	10	1	200	49	569	10	J[14], RN["User Management"]
web.dilution	10	1	196	39	1123	11	RN["Chemical Dilution Management"]
web.bill	12	1	156	39	532	12	J[20,23], RN["Reporting"]
web.rol	6	1	27	7	165	13	R
web.user	8	1	61	15	213	14	J[10], RN["User Management"]
web.area	12	1	153	36	2907	15	RN["District Management"]
exceptions	1	1	9	1	192	16	R
hibernate	3	1	19	8	168	17	R
listeners	1	1	25	5	174	18	R
mesages	1	1	5	0	188	19	R
pdf	1	1	25	4	152	20	J[12,23], RN["Reporting"]
validator	3	1	167	64	1191	21	J[2,3], RN["Security Management"]
views	2	1	13	3	144	22	R
xml	1	1	256	10	4297	23	J[12,20], RN["Reporting"]
servlet	4	1	245	35	9966	24	R
servlet.analysis	4	1	2957	399	54876	25	R
servlet.calibration	3	1	170	35	5188	26	R
servlet.customer	2	1	745	107	29251	27	R
servlet.generic	1	1	41	8	275	28	R
servlet.tag	1	1	132	28	451	29	R
servlet.user	2	1	404	65	5313	30	R
servlet.sitemesh	4	1	351	52	3766	31	R
servlet.area	2	1	565	76	15804	32	R
Total	234	32	8227	1657	145,748		
Mean	7.31	1.00	257.09	51.78	4554.63		
Std. deviation	9.81	0.00	523.95	80.33	10884.16		

the data defined in the data collection plan. Furthermore, the right side of Table 6 shows additional data about manual intervention in business process models. Firstly, an ID number is given to each analysis unit representing the identification of each preliminary business process model obtained after model transformation. Secondly, the action or actions carried out by business experts are annotated (see the set of manual operations in Table 3). The *Join* (J) actions detail the business process models that are merged between brackets and the Rename (RN) actions establish the new name between brackets. Moreover, the bottom part of Table 6 summarizes the information by means of the total, mean and standard deviation for each data column in Table 6.

Table 7 shows the final business process models obtained after manual intervention by the business experts. In addition, Table 7 shows data concerning the manual intervention by business experts for each specific business process model.

- Number of recovered tasks is the total number of tasks before manual intervention.
- Number of recovered relevant tasks is the number of tasks that the business experts mark as correct due to the fact that those

Table 7
Data from final business process models.

Business process model name	# Recovered tasks (RcT)	# Recovered relevant tasks (RcRvT)	# Recovered non-relevant tasks (RcNRvT)	# Non-recovered relevant tasks (NRcRvT)	Precision (RcRvT/RcRvT + RcNRvT)	Recall (RcRvT/RcRvT + NRcRvT)	F-measure (harmonic mean)	# Elements	Transf. time (ms)
Security management	69	30	39	4	0435	0882	0583	203	1566
Administration	45	26	19	7	0578	0788	0667	64	630
Chemical analysis management	80	51	29	5	0638	0911	0,75	419	3974
Chemical calibration management	39	28	11	6	0718	0824	0767	177	1184
User management	64	16	48	3	0250	0842	0386	261	782
Chemical dilution management	39	16	23	8	0410	0667	0508	196	1123
Reporting	53	38	15	6	0717	0864	0784	437	4981
District management	36	18	18	2	0500	0900	0643	153	2907
Total	425	223	202	41	4245	6677	5086	1910	17,147
Mean	53.13	27.88	25.25	5.13	0531	0835	0636	238.75	2143.38
Std. deviation	16.26	12.10	12.68	2.03	0163	0079	0139	129.34	1622.39

tasks faithfully represent one of the organization's business activities.

- Number of recovered non-relevant tasks is the difference between the total tasks and recovered relevant tasks. These tasks are removed from the business process because they do not represent one of the organization's business activities.
- Number of non-recovered relevant tasks represents the tasks added to a business process by the business experts to complete the organization's business operations.

Moreover, Table 7 shows the *Precision* and *Recall* values for each final business process after manual intervention by business experts, as well as the harmonic mean between them. These measures are calculated with the previous data in this table. Finally, Table 7 also presents the total number of business elements in each final business process as well as the total transformation time in milliseconds, which are derived from data collected in Table 6.

6.6. Analysis and interpretation

After the data has been collected by executing the proposed method with the *VillasanteLaboratory* system, this data is analyzed in order to draw conclusions. The analysis should obtain the chains of evidence that show inferences traceable from the data to the research questions in order to answer them. It should also be borne in mind that the alternative perspectives and explanations must be considered.

To respond to question *AQ1*, Fig. 14 shows the box chart for the *Precision* and *Recall* measures. The chart shows the mean values of the density distributions for the set of final business processes. The mean of the distribution of the *Precision* measure (0.531) is lower than the mean of the *Recall* measure (0.835). In addition, the central values of the *Recall* distribution are more concentrated around the mean than the central values of the *Precision* measure, i.e., the standard deviation of the *Recall* distribution is lower than for *Precision*.

The obtained result shows a higher *Recall* value, which means that the business archeology method recovered a higher number of relevant business elements, i.e., the proposed method recovers most of the tasks of the current business processes. However, that average value contrasts with the low *Precision* average value, which means that the number of non-relevant tasks is very high with regards to the recovered tasks. From the point of view of the business expert intervention, the results indicate that the business process archeology method obtains large business processes that should not be increased with many relevant tasks, but that the process should be reduced by removing several non-relevant tasks.

Most of the recovered non-relevant tasks are obtained almost directly from the source code and they are related to the technical nature, but they do not represent any piece of business knowledge. For instance, the obtained business process depicted in Fig. 13 has several non-relevant tasks such as *initBinder*, *doPrepareView*, *addObject*, etc., which are related to the solution domain of the system, but not to the problem domain. Unfortunately, the proposed set of patterns cannot know the different nature of each part of the source code when they are applied. Thus, this kind of non-relevant task is the main reason for the lower value of the *Precision* measure.

Despite the obtained result, this is common, since there is an inverse relationship between *Precision* and *Recall* measures [8]. Fig. 15 represents the inverse relationship between both measures, and shows the obtained results (high recall, but low precision). Ideally, the *Precision* value should be always 1.0 for any *Recall* value, but this is not possible in practice (see Fig. 15). Thus, due to the relationship between both measures has an inverse nature, it is

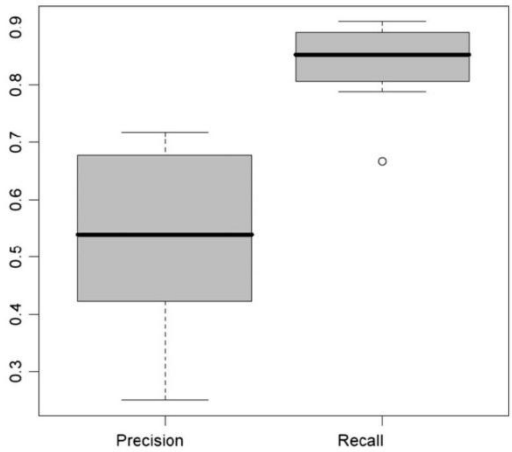


Fig. 14. The box chart for *precision* and *recall* measures.

possible to increase one only at the cost of reducing the other. Indeed, the proposed method could increase its *Precision* value by recovering fewer tasks since the precision measure has the total number of recovered task in the denominator (see Section 6.2). To maintain the higher *Recall* value, the reduction should only focus on non-relevant tasks to achieve a higher *Precision* while *Recall* is as higher as possible. It is however a hard task and some relevant tasks will probably be also discarded. As a result, while *Recall* has been reduced a little bit, *Precision* has been increased much more regarding the reduction of *Recall*. This hypothetical result would be more desirable than the obtained result, since the *Precision* and *Recall* values would be more balanced.

Since there is an inverse relationship between the *Precision* and *Recall* measures, these measures are usually not evaluated in an isolated manner. Rather, the measures are combined into a single measure known as *F-measure* [24], which consists of a weighted harmonic mean of both measures (see following equation).

$$FMeasure = \frac{2}{\frac{1}{PRECISION} + \frac{1}{RECALL}} \quad (3)$$

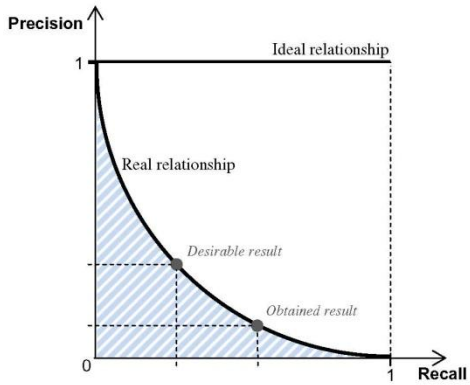


Fig. 15. Relationship between *precision* and *recall* measures and obtained results.

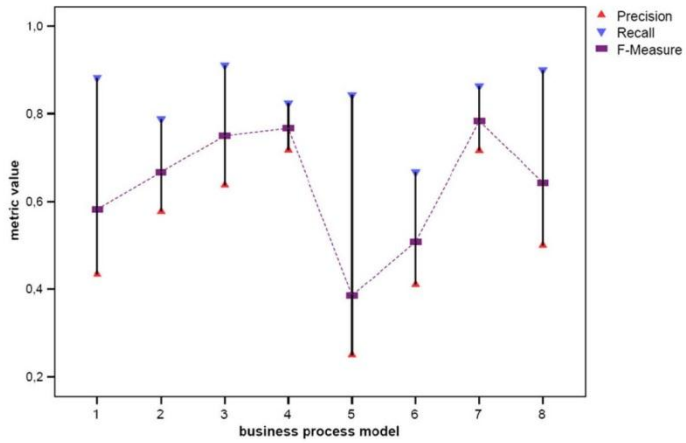


Fig. 16. F-measure values for the final business processes.

Table 7 also shows the values for the *F-Measure*, and Fig. 16 shows these values graphically with regard to the *Precision* and *Recall* values for each final business process. The *F-measure* has a mean of 0.636 and a standard deviation of 0.139. Indeed, the line of the *F-Measure* values (see Fig. 16) is clearly plotted at the top of the chart except for one business process: *5-User Management*. The value obtained in this process can be considered an outlier value, which is derived from another outlier value of the *Precision* distribution (see Table 7).

The obtained results are additionally compared with reference values from other experiences with model recovery in literature, such as those of [12,27,53]. We found reports of *Precision* and *Recall* values close to 50%, and these were our benchmark values. The average values obtained for our measures (*Precision* = 53%, *Recall* = 83% and F_1-*measure* = 64%) were therefore clearly above 50%, the reference value, and thus the results are adequate.

Therefore, question *AQ1* can be answered positively, i.e., the proposed business process archeology method makes it possible to recover business processes from legacy information systems with adequate levels of accuracy. Nevertheless, the obtained *F-Measure* value is not the best value, since it could be improved by obtaining more balanced *Precision* and *Recall* values.

Additionally, the transformation time was analyzed to answer question *AQ2*. After applying the proposed recovery process, 32 preliminary business process models were obtained with an average size of 257.1 elements (or 51.8 tasks) per business process diagram (see Table 6). The total time was 145.7 s (approximately 2.5 min), making the processing rate equal to 1.76 elements

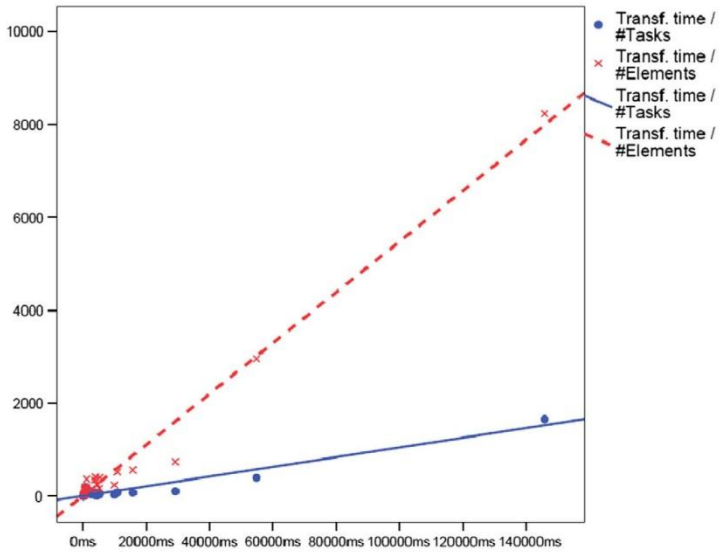


Fig. 17. The size/time scatter chart.

per second. Thus, the time spent on processing each task was 2.81 s.

The average time for each final business process model was 2.1 s (see Table 7). This time is calculated by scoring only the time value of relevant processes preliminarily recovered. However, the average time is 18.2 s if the total transformation time is considered, i.e., taking into account the time spent on recovering the non-relevant business processes as well.

The obtained result for the transformation time seems feasible for the proposed business process archeology method. Nevertheless, another derived question must be answered: is the proposed method usable for large legacy systems? In order to answer this question, Fig. 17 shows the size/time scatter chart, which reports a linear relationship between the model size and the time spent on model transformations. Since the interpolated size/time relationship is not exponential, the increase in time for large systems would be controllable.

Therefore, question *AQ2* can also be answered positively. In conclusion, the proposed recovery process makes it possible to obtain accurate business process models, which are also obtained efficiently. Thus, the main research question *MQ* is finally assessed positively.

6.7. Validity evaluation

The validity of the case study denotes the reliability of the obtained results and indicates whether the results are true and not biased for the population, i.e., the results have an adequate validity if they are valid for the whole potential population. In this study, the population is legacy information systems.

This section shows the threats to the validity of this case study. According to [52] there are mainly three types of validity (three threat categories): internal, construct and external validity.

6.7.1. Internal validity

There is no large population that makes it possible to obtain statistically representative results. However, a clear trend for the proposed measures was identifiable in this case study. In the future, we hope to contrast the result of this case with the results obtained for the same study of other legacy information systems by means of meta-analysis.

Moreover, there are two determining factors in obtaining the results presented here. On the one hand, the tool developed to support the proposed method, and used to obtain the business processes, is a factor to be taken into account when measuring the time spent on model transformations. The results could be different if the business processes are obtained with the implementation of another tool. And on the other hand, the results for the *Precision* and *Recall* measures could be also biased, due to manual intervention by business experts, since they provide their subjective opinion about the obtained business processes.

6.7.2. Construct validity

The measures in the case study were adequate to measure the variables and answer the research questions appropriately. The *Precision* and *Recall* measures were reused from the information retrieval field, where these metrics have an adequate maturity level. In addition, these measures allow us to check whether the obtained business processes faithfully represent (or not) the organization's business operations. Moreover, the time and size measures allow us to answer the research question about the scalability of the proposed business process archeology method.

6.7.3. External validity

External validity concerns the generalization of the results. In this case study, the obtained results could be generalized to legacy information systems. However, the specific platform of the selected case could be a potentially important threat. Thus, a pessimistic approach would say that the results can only be extended to legacy information systems based on Java language. In the future, we hope to replicate this study considering legacy systems based on other platforms or languages in order to compare the obtained results.

Furthermore, another important challenge is the generalization of the study's results to MARBLE. Unfortunately, it is not possible currently, since the study should be replicated several times with different business process archeology methods framed in the generic framework using different reverse engineering techniques (e.g. dynamic analysis, program slicing, etc.); different software artifacts as knowledge sources (e.g. database, user interfaces, event logs, etc.); and the combination of them. However, the result generalization to MARBLE is not so important like the future validations for each specific methods framed in MARBLE, since it is a generic and extensible framework that may be specialized in a vast number of configurations.

7. Conclusions and future work

This paper defines a business process archeology method framed in MARBLE, a generic framework to rebuild business processes underlying legacy information systems. The proposed method makes it possible to recover business knowledge from legacy information systems in order to preserve it, which can be used to modernize and maintain the legacy systems as well as to preserve the alignment between an organization's business logic and information system. Therefore, business process archeology facilitates the evolutionary maintenance of legacy systems and helps maintainers to deal with the negative effects of software ageing problems.

MARBLE, the business process recovery framework shown in this paper, is an ADM-based framework that carries out reverse engineering processes in order to progressively build business process models through four abstraction levels until it obtains business process models. Therefore, the three transformations of our proposed method are established between those levels:

- Firstly, the software artifacts of the legacy system are transformed into PSM models by statically analyzing the legacy source code.
- Secondly, the PSM models are integrated into a KDM model (PIM model) through a model transformation implemented using QVT.
- Finally, the business process models are obtained from the KDM model by means of model transformations based on pattern matching; additionally, business experts may assist at the end of this step.

Other frameworks in the literature deal with the challenge of business process archeology. However, the method proposed in this paper has three advantages over other proposals, because it is based on ADM and uses the KDM standard: (i) it automates the recovery process, leading to a reduction in maintenance costs and the extension of the legacy system lifespan; (ii) the business knowledge is managed in an integrated and standardized way through KDM; and finally (iii) the feature location is improved, which is essential for facilitating maintenance activity.

Moreover, the proposed method provides a valuable advantage regarding other alternatives like business process redesign by business experts from scratch. Our proposal allows business experts to have a better understanding about the business processes of the organization that can be refined and improved by them. Business

process redesign from scratch is slow and expensive, but in addition this solution might ignore the valuable business knowledge embedded in legacy information systems as a consequence of uncontrolled maintenance over time.

The proposed business process archeology method is also aided by a tool based on the Eclipse framework. This tool makes it possible to carry out all the stages of the proposed method including modifications of the preliminary business process models by business experts.

The proposed business process archeology method has been validated by means of a case study concerning a real-life legacy system. This case study was conducted following the case study methodology proposed by Brereton et al. The use of this methodology made the case study more rigorous, and thus increased its validity and made it easily replicable in the future. The main conclusion drawn from the case study is that the proposed method makes it possible (through the developed tool) to obtain semi-automatically business processes with an adequate level of accuracy, i.e., the resulting business process models faithfully represent the business operation of the organization that owns the system. In addition, those models are obtained in linear time with respect to the size of the models, meaning that the method is scalable to large legacy systems.

The future extensions of this research will focus on two lines. The first line consists of improving the business process archeology method. In this respect, clustering techniques will be introduced in order to reduce the size of the business process models by grouping fine-grained tasks into blocks, removing redundancies, and so on. This improvement will allow the process to obtain more complete models. This fact, in turn, will reduce the needed manual intervention by business experts, which introduces an unwanted subjective component into the method.

The second line will address the question of how to improve the validation of the proposed process. For this purpose, another case study with multiple cases is going to be carried out in order to analyze several legacy systems with different natures, platforms, and so on. These case studies may help to detect new business structure needs that would make it possible to define more refined patterns to improve the proposed method. Furthermore, variations of the proposed method will be implemented and validated by means of a set of case studies involving the same systems. The objective is to carry out a formal comparison between different business process archeology methods that consider, for instance, dynamic analysis instead of static analysis, legacy databases together source code instead of solely source code, etc.

Acknowledgments

This work was supported by the Spanish FPU Programme, and by the R&D Projects ALTAMIRA (JCCM, PII2I09-0106-2463), MOTERO (JCCM and FEDER, PEII11-0366-9449), MEDUSAS (CDTI (MICINN), IDI-20090557) and PEGASO/MAGO (TIN2009-13718-C02-01).

References

[1] W.M.P.v.d. Aalst, A.H.M.t. Hofstede, B. Kiepuszewski, A.P. Barros, Workflow patterns, Distributed and Parallel Databases 14 (3) (2003) 5–51.
[2] Archaeological Institute of America, AIA Web Portal. <http://www.archaeological.org/>, 2010 (01.28.10).
[3] P. Brereton, B. Kitchenham, D. Budgen, Z. Li, Using a protocol template for case study planning, in: Evaluation and Assessment in Software Engineering (EASE'08), 2008, Bari, Italia, pp. 1–8.
[4] Z. Cai, X. Yang, W. Wang, Business process recovery for system maintenance – an empirical approach, in: 25 th International Conference on Software Maintenance (ICSM'09), 2009, IEEE CS, Edmonton, Canada, pp. 399–402.
[5] G. Canfora, M. Di Penta, New frontiers of reverse engineering, in: 2007 Future of Software Engineering, IEEE Computer Society, 2007.
[6] B. Cornelissen, A. Zaidman, A.v. Deursen, L. Moonen, R. Koschke, A systematic survey of program comprehension through dynamic analysis, IEEE Transactions on Software Engineering 35 (5) (2009) 684–702.
[7] A. Daga, S.d. Cesare, M. Lycett, C. Partridge, An ontological approach for recovering legacy business content, Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) – Track 8, vol. 8, IEEE Computer Society, 2005, pp. 224–232.
[8] J. Davis, M. Goadrich, The relationship between precision-recall and ROC curves, in: Proceedings of the 23rd International Conference on Machine Learning, ACM, Pittsburgh, Pennsylvania, 2006, pp. 233–240.
[9] C. Di Francescomarino, A. Marchetto, P. Tonella, Reverse engineering of business processes exposed as web applications, in: 13th European Conference on Software Maintenance and Reengineering (CSMR'09), IEEE Computer Society, Fraunhofer IESE, Kaiserslautern, Germany, 2009, pp. 139–148.
[10] T. Eisenbarth, R. Koschke, D. Simon, Locating features in source code, IEEE Transactions on Software Engineering 29 (3) (2003) 210–224.
[11] EMF, Eclipse Modeling Framework Project. The Eclipse Foundation, IBM Corporation, 2009 <http://www.eclipse.org/modeling/emf/>.
[12] V.C. Garcia, D. Lucrédio, F.A. Durão, E.C.R. Santos, E.S.d. Almeida, R.P.d.M. Fortes, S.R.d.L. Meira, From specification to experimentation: a software component search engine architecture, in: 9th International Symposium on Component-Based Software Engineering (CBSE 2006), Springer-Verlag, Västerås, Sweden, 2006, pp. 82–97.
[13] A. Ghose, G. Koliadis, A. Chueng, Process Discovery from Model and Text Artefacts, in: IEEE Congress on Services (Services'07), 2007, Salt Lake City, UT.
[14] C.W. Günther, W.M.P. van der Aalst, A generic import framework for process event logs, in: Business Process Intelligence Workshop (BPI'06), 2007, LNCS 4103, pp. 81–92.
[15] W.-J.v.d. Heuvel, Aligning Modern Business Processes and Legacy Systems: A Component-Based Perspective (Cooperative Information Systems), The MIT Press, 2006.
[16] A. Hunt, D. Thomas, Software archaeology, IEEE Software 19 (2) (2002) 20–22.
[17] ikv++, Medini QVT, <http://www.ikv.de/index.php?option=com_content&task=view&id=75&Itemid=77.2008>, ikv++ technologies ag.
[18] J.E. Ingvaldsen, J.A. Gulla, Preprocessing support for large scale process mining of SAP transactions, in: Business Process Intelligence Workshop (BPI'07), 2008, LNCS 4928, pp. 30–41.
[19] ISO/IEC, ISO/IEC 14764:2006, Software Engineering – Software Life Cycle Processes – Maintenance, <http://www.iso.org/iso/catalogue_detail.htm?csnumber=39064> 2006, ISO/IEC.
[20] ISO/IEC, ISO/IEC DIS 19506. Knowledge Discovery Meta-model (KDM), v1.1 (Architecture-Driven Modernization), <http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2= 080&ics3=&csnumber=32625>, 2009, ISO/IEC, p. 302.
[21] J. Jeston, J. Nelis, T. Davenport, Business Process Management: Practical Guidelines to Successful Implementations, second ed., Butterworth-Heinemann (Elsevier Ltd.), NV, USA, 2008. p. 469.
[22] R. Kazman, S.G. Woods, S.J. Carrière, Requirements for integrating software architecture and reengineering models: CORUM II, in: Proceedings of the Working Conference on Reverse Engineering (WCRE'98), IEEE Computer Society, 1998.
[23] V. Khusidman, W. Ulrich, Architecture-Driven Modernization: Transforming the Enterprise, DRAFT V.5, <http://www.omg.org/docs/admtf/07-12-01.pdf>, 2007, OMG, pp. 7.
[24] N. Kiyavitskaya, N. Zeni, L. Mich, J.R. Cordy, J. Mylopoulos, Text mining through semi automatic semantic annotation, Practical Aspects of Knowledge Management (Lecture Notes in Computer Science), 2006, LNCS 4333, pp. 143–154.
[25] J. Koskinen, J.J. Ahonen, H. Sivula, T. Tilus, H. Lintinen, I. Kankaanpää, Software modernization decision criteria: an empirical study, in: European Conference on Software Maintenance and Reengineering, IEEE Computer Society, 2005.
[26] G.A. Lewis, D.B. Smith, K. Kontogiannis, A Research Agenda for Service-Oriented Architecture (SOA): Maintenance and Evolution of Service-Oriented Systems, 2010, Software Engineering Institute, p. 40.
[27] D. Lucrédio, R.P.M. Fortes, J. Whittle, MOOGLE: a model search engine, in: 11th International Conference on Model Driven Engineering Languages and Systems, Springer-Verlag, Toulouse, France, 2008. pp. 296–310.
[28] T. Mens, Introduction and Roadmap: History and Challenges of Software Evolution Software Evolution, Springer, Berlin, Heidelberg, 2008. pp. 1–11.
[29] T. Mens, S. Demeyer, Software Evolution, Springer-Verlag, Berlin, Heidelberg, 2008.
[30] Merriam-Webster, in: Dictionary and Thesaurus – Merriam-Webster Online, 2010.
[31] J. Miller, J. Mukerji, MDA Guide Version 1.0.1, <www.omg.org/docs/omg/03-06-01.pdf>, 2003, OMG, 62.
[32] B. Moyer, Software archeology. Modernizing old systems, Embedded Technology Journal 1 (2009) 1–4.
[33] H.A. Müller, J.H. Jahnke, D.B. Smith, M.-A. Storey, S.R. Tilley, K. Wong, Reverse engineering: a roadmap, in: Proceedings of the Conference on The Future of Software Engineering, ACM, Limerick, Ireland, 2000.
[34] OMG, Why do we Need Standards for the Modernization of Existing Systems? 2003, OMG ADM Task Force.
[35] OMG, XML Metadata Interchange, MOF 2.0/XMI Mapping, v2.1.1. <http://www.omg.org/spec/XMI/2.1.1/PDF>, 2007, OMG.

[36] OMG, QVT, Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, <http://www.omg.org/spec/QVT/1.0/PDF>, 2008, OMG.
[37] OMG, Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM), v1.1, <http://www.omg.org/spec/KDM/1.1/PDF/>, 2009, OMG, p. 308.
[38] OMG, Business Process Model and Notation (BPMN) 2.0, 2009, Object Management Group, p. 496.
[39] Open Source Initiative, JavaCC 4.2. A Parser/Scanner Generator for Java, 2009 <https://javacc.dev.java.net/>.
[40] B. Paradauskas, A. Laurikaitis, Business knowledge extraction from legacy information systems, Journal of Information Technology and Control 35 (3) (2006) 214–221.
[41] R. Pérez-Castillo, I. García-Rodríguez de Guzmán, O. Ávila-García, M. Piattini, MARBLE: a modernization approach for recovering business processes from legacy systems, in: International Workshop on Reverse Engineering Models from Software Artifacts (REM'09), 2009, Simula Research Laboratory Reports, Lille, France, pp. 17–20.
[42] R. Pérez-Castillo, I. García-Rodríguez de Guzmán, O. Ávila-García, M. Piattini, Business process patterns for software archeology, in: 25th Annual ACM Symposium on Applied Computing (SAC'10), ACM, Sierre, Switzerland, 2010, pp. 165–166.
[43] R. Pérez-Castillo, I. García-Rodríguez de Guzmán, I. Caballero, M. Polo, M. Piattini, PRECISO: a reengineering process and a tool for database modernisation through web services, in: 24th Annual ACM Symposium on Applied Computing (SAC'09), 2009, Hawaii, USA, pp. 2126–2133.
[44] R. Pérez-Castillo, I. García-Rodríguez de Guzmán, M. Piattini, Implementing business process recovery patterns through QVT transformations, in: International Conference on Model Transformation (ICMT'10), Springer-Verlag, Málaga, Spain, 2010, pp. 168–183.
[45] V. Raghavan, P. Bollmann, G.S. Jung, A critical investigation of recall and precision as measures of retrieval system performance, ACM Transactions on Information Systems 7 (3) (1989) 205–229.
[46] G. Rowe, G. Wright, The Delphi technique as a forecasting tool: issues and analysis, International Journal of Forecasting 15 (4) (1999) 353–375.
[47] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering 14 (2) (2009) 131–164.
[48] W.M. Ulrich, P.H. Newcomb, Information Systems Transformation, Architecture Driven Modernization Case Studies, Morgan Kauffman, Burlington, MA, 2010. p. 429.
[49] G. Visaggio, Ageing of a data-intensive legacy system: symptoms and remedies, Journal of Software Maintenance 13 (5) (2001) 281–308.
[50] X. Wang, J. Sun, X. Yang, Z. He, S. Maddineni, Business rules extraction from large legacy systems, in: Proceedings of the Eighth Euromicro Working Conference on Software Maintenance and Reengineering (CSMR'04), IEEE Computer Society, 2004.
[51] M. Weske, Business Process Management: Concepts, Languages, Architectures, Springer-Verlag, Berlin, Heidelberg, Leipzig, Alemania, 2007. p. 368.
[52] C. Wohlin, P. Runeson, M. Höst, O. Magnus, B. Regnell, A. Wesslén, Experimentation in Software Engineering: An Introduction, Kluwer Academic Publishers, 2000. p. 204.
[53] Y. Ye, G. Fischer, Supporting reuse by delivering task-relevant and personalized information, in: 24th International Conference on Software Engineering, ACM, Orlando, Florida, 2002, pp. 513–523.
[54] R.K. Yin, Case Study Research. Design and Methods, third ed., Sage, London, 2003.
[55] U. Zdun, C. Hentrich, S. Dustdar, Modeling process-driven and service-oriented architectures using patterns and pattern primitives, ACM Transactions on the Web 1 (3) (2007) 14.
[56] L. Zhao, L. Macaulay, J. Adams, P. Verschueren, A pattern language for designing e-business architecture, Journal of Systems and Software 81 (8) (2008) 1272–1287.
[57] Y. Zou, T.C. Lau, K. Kontogiannis, T. Tong, R. McKegney, Model-driven business process recovery, in: Proceedings of the 11th Working Conference on Reverse Engineering (WCRE 2004), IEEE Computer Society, 2004, pp. 224–233.

3.2. Static Approach

This section presents a business process archeology technique with which to discover and retrieve business processes from legacy source code. This technique is framed in MARBLE and statically analyzes source code in order to extract insights concerning business processes embedded in source code (see Figure 10). The feasibility of this technique is demonstrated with an industrial case study involving an e-government system.

This section is presented through a journal paper published by the Software: Practice and Experience Journal in 2011. This journal had an impact factor of 0.573 in 2010, according to Thomson's Science Citation Index (SCI). The reference of this paper is: *Pérez-Castillo, R., I. García Rodríguez de Guzmán, M. Piattini, and Á.S. Places, A Case Study on Business Process Recovery using an E-Government System. Software Practice & Experience Journal, 2011.42(2): p. 159–189.*

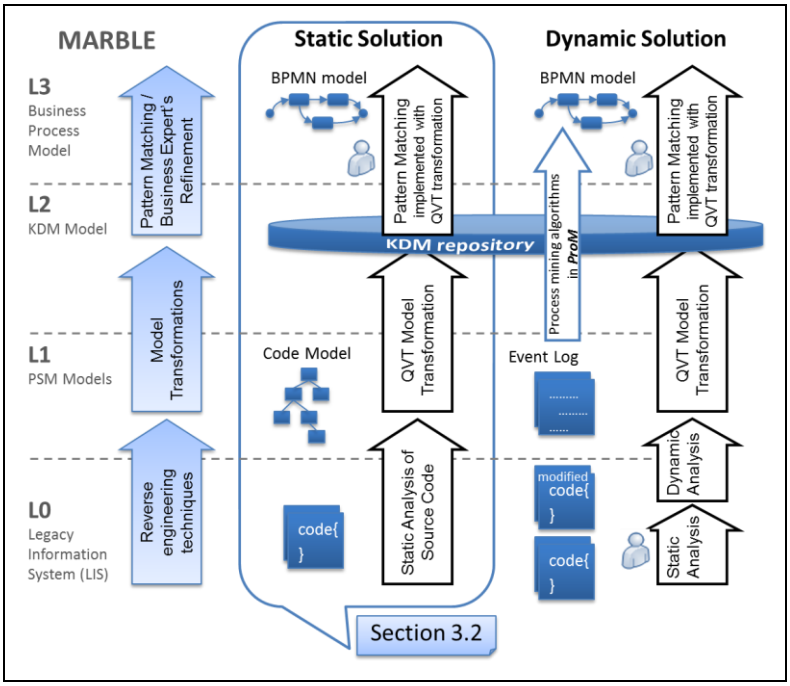


Figure 10. Main contribution of Section 3.2 as regards the proposal of the PhD Thesis as a whole

SOFTWARE – PRACTICE AND EXPERIENCE
Softw. Pract. Exper. (2011)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.1057

A case study on business process recovery using an e-government system

Ricardo Pérez-Castillo[1,*,†], Ignacio García-Rodríguez de Guzmán[1], Mario Piattini[1] and Ángeles S. Places[2]

[1]*Alarcos Research Group, University of Castilla-La Mancha, Paseo de la Universidad, 4 13071 Ciudad Real, Spain*
[2]*Database Lab., Universidade da Coruña, Facultade de Informática, Campus de Elviña s/n, 15071 A Coruña, Spain*

SUMMARY

Business processes have become one of the key assets of organization, since these processes allow them to discover and control what occurs in their environments, with information systems automating most of an organization's processes. Unfortunately, and as a result of uncontrolled maintenance, information systems age over time until it is necessary to replace them with new and modernized systems. However, while systems are aging, meaningful business knowledge that is not present in any of the organization's other assets gradually becomes embedded in them. The preservation of this knowledge through the recovery of the underlying business processes is, therefore, a critical problem. This paper provides, as a solution to the aforementioned problem, a model-driven procedure for recovering business processes from legacy information systems. The procedure proposes a set of models at different abstraction levels, along with the model transformations between them. The paper also provides a supporting tool, which facilitates its adoption. Moreover, a real-life case study concerning an e-government system applies the proposed recovery procedure to validate its effectiveness and efficiency. The case study was carried out by following a formal protocol to improve its rigor and replicability. Copyright © 2011 John Wiley & Sons, Ltd.

Received 22 October 2010; Revised 29 December 2010; Accepted 30 December 2010

KEY WORDS: business process mining; legacy system; ADM; model transformation; case study

1. INTRODUCTION

Business processes are increasingly becoming essential assets for organizations since they reflect owner organizations' logic of 'how to'. Weske [1] states that a business process depicts a set of activities performed in an organization that jointly realize a business goal. These descriptions provide a means to map business objectives regarding how to best carry out operations within organizations. Therefore, business processes also assist in the process of getting automated information systems to achieve these business objectives [2]. Most organizations' business processes are currently supported by their enterprise information systems. Indeed, according to Heuvel [3], business processes are usually the starting point for developing information systems as a part of the requirement analysis.

However, according to Visaggio [4], enterprise information systems are not immune to software erosion and software aging. These systems become progressively less maintainable over time as a result of uncontrolled maintenance. The immediate effect of this maintenance is that the system

*Correspondence to: Ricardo Pérez-Castillo, Alarcos Research Group, University of Castilla-La Mancha, Paseo de la Universidad, 4 13071 Ciudad Real, Spain.
†E-mail: ricardo.pdelcastillo@uclm.es

Copyright © 2011 John Wiley & Sons, Ltd.

maintainability diminishes below the acceptable limits. The information system therefore becomes a *legacy* information system and must be replaced with a new one.

Nevertheless, when organizations replace or update an enterprise information system, business processes do not reflect changes occurring during the maintenance stage. Indeed, very often the business processes documentation is neither updated nor documented at all [5]. Therefore, aging or obsolete legacy information systems should not be entirely discarded because they might contain a lot of latent meaningful business knowledge as a consequence of modifications over time [6]. This knowledge is embedded in the system and it might not, therefore, be present anywhere else. Thus, according to [7], when a legacy system is replaced with another improved system, business knowledge preservation is an important challenge that must be addressed.

Business knowledge preservation can be carried out by means of updating the system documentation, but the implicit business processes must also be recovered from the legacy system. The recovered business processes can then be used by organizations in two ways: (i) to provide business experts with a better understanding of the real, current operation of the organization and (ii) if necessary, to develop the new improved system in order to mitigate the effects of the software aging and erosion. The evolved system will thus support the current business processes and will also improve the ROI (*Return Of Investment*) of the legacy system, since it extends its lifespan.

This paper deals with the problem of business process recovery from legacy information systems in order to address the problem of business knowledge preservation. This is not a new problem, and reverse engineering applied to the recovery of business logic from legacy systems has been frequently addressed by business experts and software academics [8].

Reengineering has normally been used to obtain new improved versions of aged legacy systems and, according to [9], this consists of three stages: reverse engineering, restructuring, and forward engineering. The reverse engineering stage can obtain abstract representations of the system, including the representation of business knowledge, which is then used in restructuring and forward engineering stages. However, Khusidman and Ulrich [10] state that reengineering projects rarely reach the business abstraction level, and typically reach only the system design level. According to [11], reengineering additionally lacks in formalism and standardization. Indeed, the majority of reengineering projects are usually carried out in an *ad hoc* manner.

Software modernization helps to solve the formalism and standardization problems of reengineering. Software modernization, and in particular ADM (Architecture-Driven Modernization) as defined by the OMG (Object Management Group) [12], advocates carrying out reengineering processes by considering model-driven development principles. ADM-based processes deal with all involved artifacts as models that conform to specific metamodels, and these processes also establish model transformations between models to deal with different abstraction levels throughout the three reengineering stages [13]. ADM-based processes therefore facilitate business process recovery. This is owing to the fact that these processes appropriately manage the large conceptual gap between legacy systems and business processes through the definition of models at different abstraction levels and incremental transformations between those levels.

As a solution to the aforementioned problem, we introduced MARBLE (*Modernization Approach for Recovering Business processes from LEgacy systems*) in a previous work [14], which is a generic framework that implements how to use ADM to recover business knowledge from legacy information systems. The objective of MARBLE is to provide the highest abstraction level during the reverse engineering stage of ADM, i.e. the business knowledge that depicts the information systems. The MARBLE framework provides a solution to meet the demands detected by Khusidman and Ulrich [10]. Moreover, it is aligned with the SOA (Service-Oriented Architecture) research agenda developed by the SEI (Software Engineering Institute) [15], which reports that business process recovery is needed for modernizing legacy information systems toward SOA systems.

This paper presents an entire functional business process recovery procedure framed in MARBLE, which is specially implemented for object-oriented systems. The proposed procedure is aligned with the four different abstraction levels proposed in MARBLE: L0 to represent the legacy system; L1, which contains different platform-specific models (PSM) to depict different software artifacts of the legacy system; L2, which integrates all the specific L1 models into

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

a platform-independent model (PIM), which is represented according to KDM (Knowledge Discovery Metamodel) [16]; and finally L3 to represent the business process model.

Furthermore, the proposed procedure for recovering business processes defines three model transformations between the four levels: (i) legacy source code is considered as the key software artifact in L0, and static analysis as the reverse engineering technique used to extract meaningful knowledge from L0 and represent it in L1; (ii) a model transformation based on QVT (Query/Views/Transformations) to obtain a KDM model in L2 from code models in L1; and finally (iii) a set of business patterns supported by means of a QVT transformation [17] to transform the KDM modes in L2 into business process models in L3.

The last transformation can be additionally supported by the manual intervention of business experts to refine the business processes obtained (e.g. by adding manual tasks that are not supported by information systems). Thus, we propose a semiautomatic procedure for recovering business processes. This can be considered as a critical point of our proposal, since it requires an additional manual intervention by experts. However, the proposed procedure has two main advantages compared with the business process redesign by experts from scratch. First, redesign from scratch is a more time-consuming and harder option than our proposal, which provides an initial meaningful understanding of current business processes. Second, the manual business process redesign by business experts does not consider the business knowledge embedded in legacy information systems, which is not present in any other artifact.

In order to automate the approach and to facilitate its adoption, we also provide a tool especially developed to support the proposed procedure for recovering business processes. This tool also allows business experts to modify business processes at the end of the procedure. Moreover, the tool has facilitated the conduction of an industrial case study involving a real-life information system in order to empirically validate the proposal. The case study has been conducted by following the formal protocol for case studies proposed by Brereton *et al.* [18] in order to improve the rigor and validity of the study.

The case study involves an e-government system of the Spanish electronic administration. The case study consists of the recovery of the embedded business processes, and it then evaluates the effectiveness and efficiency of the proposed procedure framed in MARBLE. The effectiveness of the procedure is validated in terms of whether the business processes recovered from the legacy system faithfully represent the organization's business behavior. Moreover, in order to validate the efficiency of the proposal, the study evaluates the time spent on recovering the business processes with regard to the size of the system. After result analysis, the case study reports that the procedure enables business process recovery with adequate accuracy levels. The study also concludes that the proposed framework is scalable to larger systems.

The remainder of this paper is organized as follows. Section 2 provides some notions to have a better understanding of our proposal. Section 3 summarizes the works related to the business process recovery and compares them with our proposal. Section 4 presents the proposed procedure for recovering business processes embedded in legacy information systems. Section 5 summarizes the most relevant details concerning tool implementation to support the proposed procedure. Section 6 presents the planning and execution of the case study that involves a real-life e-government system. Finally, Section 7 discusses conclusions and future work.

2. BACKGROUND

The following subsections introduce the two main concepts to have a better comprehension of the proposal: the ADM approach as well as the KDM standard.

2.1. ADM

The increasing cost of maintaining legacy systems, together with the need to preserve business knowledge, has turned the modernization of legacy systems into a significant research field [19].

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

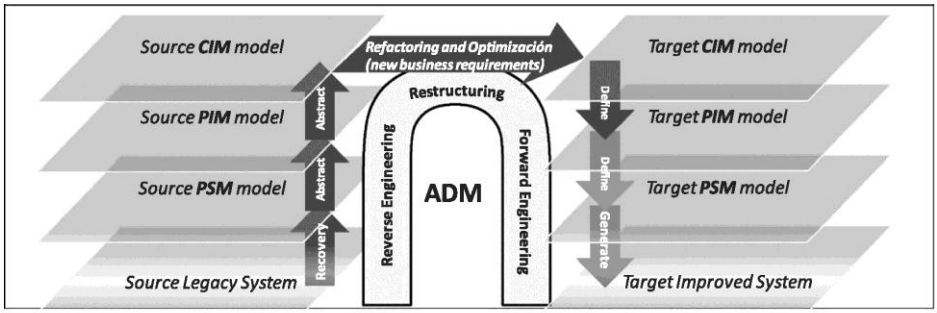


Figure 1. Horseshoe modernization model.

ADM can be considered as a mechanism for software evolution, i.e. it makes it possible to modernize legacy information systems and eradicates, or at least minimizes the negative effects of the software aging phenomenon in legacy systems. According to [20], ADM is the process of understanding and evolving existing software assets, and in addition, it restores the value of existing applications. ADM is based on reengineering, but it considers different models as input and output artifacts of the process, thus solving the formalization and automation problem found in traditional reengineering.

The traditional horseshoe reengineering model [21] was adapted to ADM, which became the horseshoe modernization model (see Figure 1). This model consists of three stages:

- *Reverse engineering* is represented by the left side of the horseshoe. It analyzes the legacy system in order to identify the components of the system and their interrelationships. In turn, the reverse engineering stage builds one or more representations of the legacy system at a higher level of abstraction.
- *Restructuring* is represented by the curve of the horseshoe since this stage takes the previous system's representation and transforms it into another one at the same abstraction level. This stage preserves the external behavior of the legacy system.
- *Forward engineering* is represented by the right side of the horseshoe because it generates physical implementations of the target system at a low abstraction level from the previously restructured representation of the system.

Moreover, the horseshoe modernization model considers three different kinds of models with respect to the abstraction level [22]:

- *Computation-independent model (CIM)* is a business view of the system from a computation-independent viewpoint at a high abstraction level. CIM models are sometimes called domain models.
- *PIM* is a view of a system from the platform-independent viewpoint at an intermediate abstraction level. PIM models abstract all implementation details.
- *PSM* is a technological view of a system from the platform-specific viewpoint at a low abstraction level. A PSM combines the specifications in the PIM with the details that specify how that system uses a particular type of platform or technology.

Transformations between the different kinds of models are formalized by means of the QVT standard proposed by the OMG [23]. The QVT specification consists of two distinct but related languages: (i) *QVT-Operational* language, which is procedural in nature, and (ii) *QVT-Relations*, a declarative language. QVT makes it possible to define deterministic transformations between models at the same abstraction level or at a different level. As a consequence, the model transformations can be automated in the horseshoe modernization model.

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

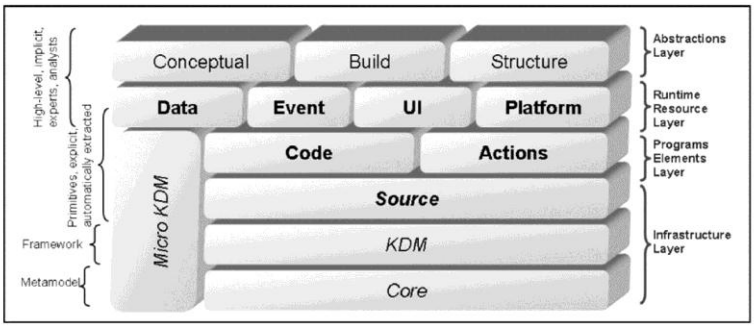


Figure 2. Layers, packages, and concerns in KDM (adapted from [16]).

2.2. *KDM*

In addition, the ADM Task Force in the OMG has defined the KDM standard [24], which has also been recognized as the ISO 19506 standard [16]. The KDM standard defines a metamodel for modeling all the different legacy software artifacts involved in a legacy information system. The KDM metamodel provides a comprehensive high-level view of the behavior, structure, and data of the legacy systems [16], but it does not represent procedural models of the systems such as UML (Unified Modeling Language). While UML can be used to generate new code in a *top-down* way, ADM-based processes involving KDM start from the legacy code and build a higher level model in a *bottom-up* way [25].

KDM is a metamodel divided into layers representing both physical and logical *software assets* of information systems at several abstraction levels [26]. KDM separates knowledge about legacy information systems into various orthogonal concerns that are well known in software engineering as *architecture views*. KDM consists of four layers of abstraction, each one based on the previous layer [16] (see Figure 2).

- *Infrastructure layer.* is the layer at the lowest abstraction level and defines a small set of concepts used systematically throughout the entire KDM specification. There are three packages in this layer: Core, KDM, and Source.
- *Program elements layer.* offers a broad set of metamodel elements in order to provide a language-independent intermediate representation for various constructs defined by common programming languages. This layer represents implementation level program elements and their associations. This means that the program elements layer represents the logical view of a legacy system. This layer has two packages: Code and Action. Both packages define a single model, called the *CodeModel*.
- *Runtime resource layer.* enables the representation of knowledge about resources managed by the legacy system's operation environment, i.e. it focuses on those things that are not contained within the code itself. It has four packages: Data, Event, UI, and Platform.
- *Abstraction layer.* defines a set of metamodel elements whose purpose is to represent domain-specific knowledge as well as provide a business-overview of legacy information systems. This layer has three packages: Conceptual, Structure, and Build.

A very important challenge in business process recovery lies in the large conceptual gap that exists between business processes and legacy systems. This must be gradually reduced. ADM facilitates business process archeology by means of KDM, since it reduces the conceptual gap due to the fact that it is organized into several layers at different abstraction levels [16]. Thus, KDM makes it possible to carry out endogenous transformations (i.e. transformations between models concerning the same metamodel) from models in lower layers to models in higher layers of the KDM structure. Therefore, specific business knowledge is extracted directly from the legacy

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

system, at which point the implicit knowledge at a higher abstraction level can be inferred or deduced from the prior knowledge.

3. RELATED WORK

Companies and academics have had a common trend for many years to recover business knowledge embedded in information systems in order to preserve that valuable knowledge during system evolution. For this purpose, several reverse engineering techniques have been used, although the most common techniques are static and dynamic analyses. In addition, the recent advances in model-driven development research area make it possible to improve the business knowledge extraction mechanisms by means of model reuse, formalization, and standardization. Table I, and in general this section, provides a comparison between different proposals in the literature for recovering the embedded business knowledge. Table I frames each recovery procedure in a matrix of software artifacts or knowledge sources in rows, along with the mechanism used to extract the business knowledge in columns.

Some works address business knowledge recovery from legacy information systems by statically analyzing the source code. Zou *et al.* [30] developed a model-driven framework based on a set of heuristic rules for extracting business processes. This framework syntactically analyzes the legacy source code and applies the rules to transform pieces of source code in certain business process elements. Despite this work being based on the MDA approach, it does not consider ADM's standards like KDM. Since KDM facilitates the integrated representation of several heterogeneous systems, this work is not able to recover system-transversal business processes executed through more than one system. This proposal was validated by means of a case study concerning an e-commerce system, but no formal protocol to conduct the case study was used.

In addition to source code, other software artifacts are also considered to statically obtain business processes. Ghose *et al.* [31] provides a set of text-based queries executed in documentation for extracting business knowledge, although this work is not based on the MDA approach. The intent of this approach for text-to-model extraction is to look for cues within text documents that suggest some process model fragments. This proposal was only validated through an example.

System databases are other artifacts used as input in business knowledge recovery based on static analysis. Paradauskas *et al.* [32] recover business knowledge by means of the inspection

Table I. Comparison between business knowledge recovery proposals.

Extraction technique / SW artifact	Not model-driven		Model-driven	
	Static analysis	Dynamic analysis	Static analysis	Dynamic analysis
External expert information	Cai *et al.* [27]	Eisenbarth *et al.* [28]	* Proposed Business Recovery Procedure	
Source code	Wang *et al.* [29]		Zou *et al.* [30]	
Documentation	Ghose *et al.* [31]			
Database	Paradauskas *et al.* [32]		Pérez-Castillo *et al.* [33]	MARBLE (Generic framework)
User interfaces		Di Francescomarino *et al.* [34]		
Event logs		Günther *et al.* [35] Ingvaldsen *et al.* [36] van der Aalst *et al.* [37]		

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

of the data stored in databases together legacy application code. This work does not follow model-driven development principles either. The authors of this work validated their proposal through the development of a controlled example. Another work, taking database as the input artifact, is provided by Perez-Castillo *et al.* [33], which proposes a model-driven reengineering framework to extract business logic from relational database schemas. This work does not use the KDM standard, thus it has the same limitation as the Zou *et al.* work. This proposal was validated by means of a case study in which the framework was applied to a real legacy relational database. Unfortunately, this study did not use a formal protocol to conduct the case studies.

Moreover, Wang *et al.* [29] present a framework for business rules extraction from large legacy information systems based on static program slicing. Program slicing is a reverse engineering technique consisting of the program decomposition into slices according to some criteria (e.g. fragments of source code that uses a specific program variable). However, the framework is not able to represent the recovered business knowledge in an understandable and standard way. This work was validated by means of a large complex financial legacy system, but this validation did not follow a formal protocol either.

All these works solely use static analysis as the reverse engineering technique, which has the inconvenience that a lot of business knowledge is lost since it ignores all runtime knowledge. Dynamic analysis deals with this problem, which makes it possible to consider the information derived by system execution [38]. For instance, Eisenbarth *et al.* [28] present a business feature location technique based on dynamic analysis. Previously, domain experts manually defined a set of scenarios invoking features. The scenarios were then executed collecting business knowledge mapped between general and specific computational units with respect to a given set of features. This proposal does not follow model-driven principles, and in addition it does not extract business knowledge according to any business process notation. This work demonstrated its applicability by means of an example. Cai *et al.* [27] propose an approach that combines the requirement reacquisition with a dynamic and static analysis technique to extract complex business processes that are triggered by external actors. In a first step, the use cases are recovered through interviews with the users of the legacy information system. In a second step, the system is dynamically traced according to those use cases. Finally, the traces obtained in runtime are statically analyzed to recover the business processes. This work was also validated by means of a non-formal case study. Moreover, Di Francescomarino *et al.* [34] recover business processes by dynamically analyzing the Web application GUI forms which are executed during the user's navigation. This work additionally provides a clustering algorithm to minimize the obtained business processes. However, this proposal is not aligned with model-driven development principles either. This work was validated by means of a case study that involves an e-commerce system, which was similarly conducted in a non-formal way.

Other works addressing the dynamic approach provide process mining techniques that register event logs focusing on Process-Aware Information Systems (PAIS), i.e. process management systems such as Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) systems. The nature of these systems (in particular their process awareness) facilitates the registration of events throughout process execution. Event logs depict the sequence of business process' activities executed, and can be used to dynamically discover the current business processes. There is much work following this approach, for instance, van der Aalst *et al.* [37] provides a large industrial case study by applying different business process mining algorithms to discover business process form event logs. This large study was not conducted following a formal protocol. Günther *et al.* [35] also provide a generic import framework for obtaining event logs from different kinds of PAIS. In addition, Ingvaldsen *et al.* [36] focus on ERP systems to obtain event logs from the SAP's transaction data logs. Both works do not follow the model-driven development principles. These two proposals were validated, although they did not follow a formal protocol for conducting case studies.

Dynamic analysis is more powerful than static analysis, since there is specific knowledge that is known only at runtime (e.g. the specific data of the objects instantiated during execution). However, the dynamic analysis usually requires source code modifications in order to aggregate

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

traces to register system execution information in an appropriate way. However, source code modifications are not always possible since legacy information systems can be in the production stage. Moreover, another point of controversy related to the usage of dynamic analysis techniques in realistic environments is that it might be thought that a company or organization would not accept the use of an automatically modified version of its information system. For this reason, process recovery procedures based on the dynamic analysis technique should ensure that the source code modification is carried out without affecting the behavior of the original information system, i.e. in a non-invasive way.

4. BUSINESS PROCESS RECOVERY PROCEDURE

The proposed business process recovery procedure is framed in MARBLE. While MARBLE depicts a general-purpose and extensible ADM-based framework for recovering business processes (cf. Section 4.1), the concrete procedure presented in this paper establishes the three specific transformations to deal with the business process recovery challenge (cf. Sections 4.2–4.4). Specifically, this paper proposes a specific procedure based on static analysis as the main technique and focusing on object-oriented source code and manual post-intervention by business experts as the unique sources of knowledge.

4.1. MARBLE

The MARBLE framework proposed in [14] is a generic ADM-based framework. This framework provides the guidelines to use the ADM approach and their standards like KDM for recovering business knowledge from different legacy software artifacts and following different reverse engineering techniques (i.e. MARBLE is a heterogeneous framework). That heterogeneity is achieved by using the KDM standard to represent legacy information systems, which is a multi-view standard that defines a metamodel at different abstraction levels. It is not feasible to use all knowledge sources with all reverse engineering techniques at the same time. Therefore, the guidelines provided by MARBLE must be particularly implemented for each software artifact as a knowledge source, as well as for each reverse engineering technique.

Figure 3 shows the MARBLE framework, which is focused on the reverse engineering stage of the horseshoe modernization model. MARBLE specifically defines four abstraction levels related to four different kinds of models: L0 to L3. In addition, MARBLE specifies the model transformation path needed to obtain each model at a specific level from the previous one (see Figure 3).

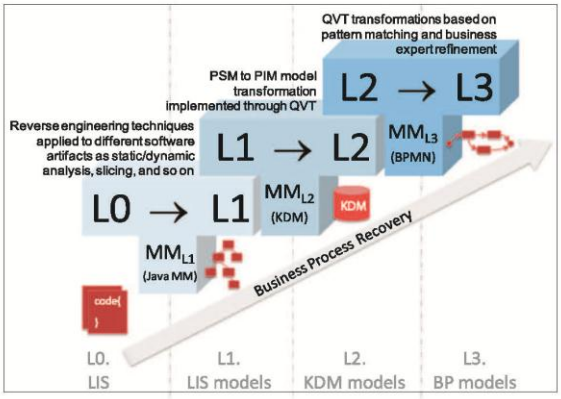


Figure 3. Overview of MARBLE.

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

- *L0*: *Legacy information system.* This level represents the entire legacy system as it is in the real world, as a set of interrelated software artifacts: source code, user interfaces, databases, documentation, and so on.
- *L1*: *Legacy information system models.* This level contains a set of PSM models that can represent one or more software artifacts of the legacy system. The models at the L1 level are represented according to specific metamodels that specify how a software artifact uses a particular type of platform or technology. For example, in a recovery procedure that follows MARBLE, a hypothetic L1 level could be formed of a code model represented according to the Java metamodel and a database model depicted according to the SQL metamodel.
- *L2*: *KDM model.* This level contains a sole PIM model that integrates all the specific PSM models of the previous L1 level. In order to represent this model, MARBLE uses the KDM metamodel, since it makes it possible to model all the artifacts of the legacy system in an integrated and technological-independent manner. First, L2 is obtained in an integrated way because L2 works as a KDM repository that can be progressively populated with knowledge extracted from the different information systems of an organization. This is due to the fact that the *structure* package at the last KDM abstraction layer (cf. Section 0) is able to represent different systems, subsystems, components, architecture views, and so on. This is a key advantage when business processes are transversely executed by multiple systems within an organization. Second, L2 is represented in a technological-independent way due to the fact that KDM standard abstract from the program element layer (the second layer of the KDM metamodel) all those details concerning the technological viewpoint (e.g. the program language). Thereby, the KDM repository at L2 can represent several legacy information systems even when their program languages or platforms are heterogeneous.
- *L3*: *Business process models.* This level corresponds to models that represent the business processes recovered from the legacy system, i.e. the model at this level is considered as a CIM model. MARBLE uses the metamodel of the BPMN (Business Process Modeling and Notation) standard [39]. BPMN offers a well-known graphical notation that is easily understood by both system analysts and business experts.

The proposed business process recovery procedure defines the three concrete transformations between these four levels. These transformations are presented in the following subsections.

4.2. L0-to-L1 transformation

The first transformation takes the different software artifacts from legacy information systems and obtains a specific model for each one. The software artifacts considered in this transformation depend on the specific business process recovery procedure framed in MARBLE. If the specific method considers more artifacts, it will probably have more sources from which to extract the business knowledge needed.

We are currently contemplating a recovery procedure framed in MARBLE, which considers legacy source code as the unique software artifact since, according to [40], it is the artifact that embeds most business knowledge. The L0-to-L1 transformation thus obtains a code model that represents the source code of the legacy system at a low abstraction level, i.e. it considers technological details. Indeed, this transformation is, in this case, tuned to analyze Java-based systems. Nevertheless, this transformation can be extended in each case by considering more program languages, which will imply more than one code model at L1, although those code models would be then integrated at L2 through the KDM model.

The analysis can be carried out by means of different reverse engineering techniques such as static analysis which examines the source code files, dynamic analysis which analyzes the source code at runtime, and program slicing which studies the code and divides it into different fragments. This transformation uses static analysis as the reverse engineering technique to extract the necessary information. The transformation is therefore supported by a parser that syntactically analyzes the Java source files of the legacy information system from a static viewpoint and builds Java code models according to a Java metamodel. In particular, the parser analyzes the text of a Java source file, considering it as a sequence of tokens (e.g. keywords, identifiers, and so on) to

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

Figure 4. An example of the L0-to-L1 transformation for a product shipping system.

determine whether the text represents a valid Java code file. The parser builds an abstract syntax tree of the Java code file on the fly, which represents a Java code model at L1.

An example will now be shown to illustrate the performance of the proposed transformations. Let us imagine an information system supporting the product shipping of a reseller organization. This example focuses on a certain piece of source code: the *ResellerControler* class of the *domain* package (see Figure 4, left side). This class contains four methods that support four key functionalities of the system: (i) *receiveOrder* manages customers' order requests; (ii) *checkInventory* checks whether the products needed to fulfill a specific order are in stock; (iii) *sendProducts* manages the dispatch of the products ordered; and finally (iv) *sendInvoice* generates and sends the invoice to the customer.

According to the example, the L0-to-L1 transformation takes the Java source file (see Figure 4, left-hand side) and obtains the abstract syntax tree representing the Java code model at L1 (see Figure 4, right-hand side). The parser obtains a *CompilationUnit* element for each of the Java source files analyzed, and then adds the *PackageDeclaration* and *ImportDeclaration* elements. The parser then generates a *ClassOrInterfaceDeclaration* element for the class, which contains a *ClassOrInterfaceBodyDeclaration* element for each method. Each method is represented through a *MethodDeclaration* with a *ResultType*, *MethodDeclarator*, and a *Block* of *Statement* elements. The *Statement* element is, in turn, specialized into several kinds of elements: *ReturnStatement*, *IfStatement*, *Expression*, and so on. The right-hand side of Figure 4 shows the elements used to represent the business logic implemented in the *receiveOrder* method.

4.3. L1-to-L2 transformation

The L1-to-L2 transformation is in charge of the transformation of the PSM models of L1 into a single PIM model in L2 according to the KDM metamodel. The KDM model in L2 considers the

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

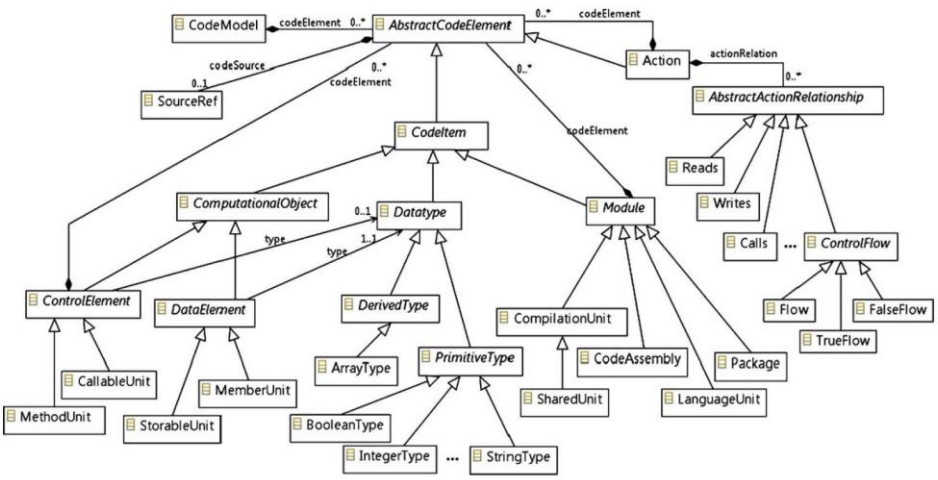


Figure 5. Overview of action and code packages of the KDM metamodel.

code and *action* KDM packages that conform to the *program element* layer of the KDM metamodel (see Figure 2). The proposal uses only these packages because legacy source code is the unique artifact considered at L0 according to our procedure design.

Figure 5 shows the most important metaclasses in the *code* and *action* packages of the KDM metamodel. According to the KDM *code* metamodel, each of the legacy systems analyzed must be represented as an instance of the *CodeModel* metaclass, which is the root metaclass. A *CodeModel* is composed of *AbstractCodeElements*, a metaclass that represents an abstract parent class for all KDM entities that can be used as *CallableUnit*, *StorableUnit*, and so on. The *CodeElement* metaclasses are also interrelated by means of *AbstractCodeRelationships* (*action* package),a metaclass representing an abstract parent for all KDM relationships that can be used to represent the code relationships, such as *Flow*, *Calls*, *Reads*, and *Writes*.

In order to establish the model transformation between the code model (the PSM model at L1) and the KDM code model (the PIM model at L2), we use QVT to define the transformation. The L1-to-L2 transformation specifically uses the QVT-Relation language, since the implementation of the transformation is easy in a declarative manner. This is owing to the fact that the structures of the Java metamodel (L1) and KDM code-action metamodel (L2) are very similar. Indeed, in some cases the transformation is as simple as renaming the metaclasses between the two metamodels, although the transformation may not be as trivial for other elements.

A QVT transformation consists of several relations focusing on the transformation of specific elements. Each relation defines at least two domains of elements: (i) the source domain, tagged as *checkonly*, which evaluates whether the specific configuration of elements of the input metamodel (Java metamodel) exists, and the target domain, tagged as *enforce*, which evaluates the configuration of elements of the output metamodel (KDM) and creates or destroys elements to satisfy the rules of the relation. A QVT relation can also have *when* or *where* clauses which establish the pre- and post-conditions of execution. Figure 6 shows a fragment of the proposed QVT transformation: the *class2compilationUnit* and *method2callableUnit* QVT relations. The *class2compilationUnit* relation transforms all the instances of the *Class* metaclass in the Java code model (L1) into instances of *CompilationUnit* in the KDM model (L2). The relation also examines the *Method* instances belonging to the *Class* instance in the where clause (see Figure 6) and the *method2callableUnit* relation is triggered. The *method2callableUnit* relation is in charge of the transformation of the instances of the *Method* metaclass into *CallableUnit* instances with the same name and type. All the different kinds of statements in the Java code model are then transformed into *codeElement* instances by means of various QVT relations triggered in the where clause.

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

```
transformation Java2KDM (java:java, kdm:code){
  ...
  relation class2compilationUnit {
      className : String;
      checkonly domain java jc : java::classifiers::Class {
          name = className
      };
      enforce domain kdm kp : KDM_MetaModel::code::Package {
          codeElement = kcu : KDM_MetaModel::code::CompilationUnit {
              name = className
          }
      };
      where {
          jc.members->forAll (jm:java::members::Method | jm.oclIsKindOf(java::members::Method)
              implies method2callableUnit (jm, kcu));
      }
  }
  relation method2cllableUnit {
      methodName : String;
      methodType : String;
      checkonly domain java jm : java::members::Method {
          name = methodName,
          typeReference = jtr : java::types::TypeReference {
              name = methodType
          }
      };
      enforce domain kdm kcu : KDM_MetaModel::code::CompilationUnit {
          codeElement = cu : KDM_MetaModel::code::CallableUnit {
              name = methodName,
              type = t : KDM_MetaModel::code::Datatype {
                  name = methodType
              }
          }
      };
      where {
          jm.parameters->forAll(jp:java::parameters::Parameter | jp.oclIsKindOf(java::parameters::
              Parameter) implies parameter2codeElement(jp, cu));
          jm.members->forAll(js:java::statements::Statement | js.oclIsKindOf(java::statements::
              ExpressionStatement) implies expressionStatement2codeElement(js, cu));
          jm.members->forAll(js:java::statements::Statement | js.oclIsKindOf(java::statements::
              Conditional) implies conditional2codeElement(js, cu));
          ...
          jm.members->forAll(js:java::statements::Statement | js.oclIsKindOf(java::statements::
              Return) implies return2codeElement(js, cu));
      }
  }
  ...
}
```

Figure 6. QVT relations to transform Java code models into KDM models.

To continue with the example, Figure 7 shows the KDM model obtained after the execution of the proposed QVT transformation for the product shipping system. The KDM model contains a *Package* instance named *domain* with a nested *CompilationUnit* instance named *Reseller*. The *CompilationUnit* is obtained from the *Class* instance of the Java code model at L1. The *Reseller CompilationUnit* instance contains an instance of the *CallableUnit* metaclass for each Java method at L1. Each *CallableUnit* instance is also defined by means of different *CodeElement* and *ActionElement* instances. For example, the *receiveOrder* method is modeled as a *CallableUnit* containing (see Figure 7): (i) an *EntryFlow* instance that defines the first KDM action in the unit (the first Java statement, since a model's sequentiality of actions is not clearly defined); (ii) a *Signature* instance that defines the parameters of the unit; and finally (iii) an *ActionElement* instance that represents the *if* statement in the Java method. The remaining *CallableUnit* instances have a similar structure.

4.4. L2-to-L3 transformation

The L2-to-L3 transformation is the last transformation and consists of two steps: (i) a model transformation that obtains a first version of business process models and (ii) manual post-intervention by business experts who modify and refine the business processes obtained in order to improve them.

First, the model transformation establishes a set of business patterns [41], which define which pieces of the source code (represented in a KDM code model) are transformed into specific

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

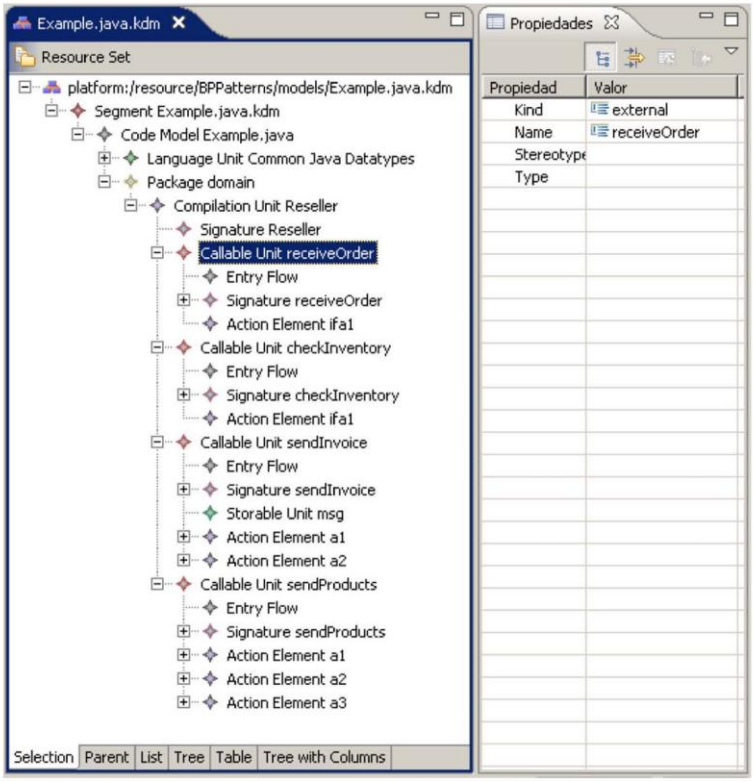


Figure 7. An example of the L1-to-L2 transformation for a product shipping system.

structures of a business process. The business process models at L3 are represented according to the BPMN metamodel (see Figure 8). The BPMN metamodel represents *business process diagrams* (BPD) that involve four kinds of elements: (i) flow object elements such as *events*, *activities*, and *gateways*; (ii) connecting object elements, such as *sequence flows*, *message flows*, and *associations*; (iii) artifact elements, such as *data objects*, *groups*, and *annotations*; and (iv) swim lane elements for grouping elements, such as *pools* and *lanes*.

The set of patterns are defined in terms of KDM and BPMN elements, and the last transformation is thus independent of the program language and platform of the legacy system in contrast to the first and second transformations. This signifies that the set of proposed patterns could also be used with other systems in addition to Java-based systems.

Table II presents the set of proposed business patterns of the L2-to-L3 transformation. The patterns take well-defined business patterns from the literature that have been successfully applied by business experts in the modeling of business processes [41–44]. These business patterns are adapted in order to consider which specific structure of the legacy system (represented in a KDM model) is transformed into each specific business pattern. Each pattern therefore consists of (i) a source configuration or structure of elements in the input model (KDM model in L2) and (ii) a target structure of elements in the output model (BPMN model in L3).

In order to support the set of patterns, a model transformation is implemented using QVT-Relations [17], which supports the pattern matching process. Each QVT relation searches for instances of the source structures defined by each pattern. The QVT relations, which are defined in a declarative manner, then enforce the creation in the business process model of instances of the target structures of the pattern for each input instance found in the KDM model.

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

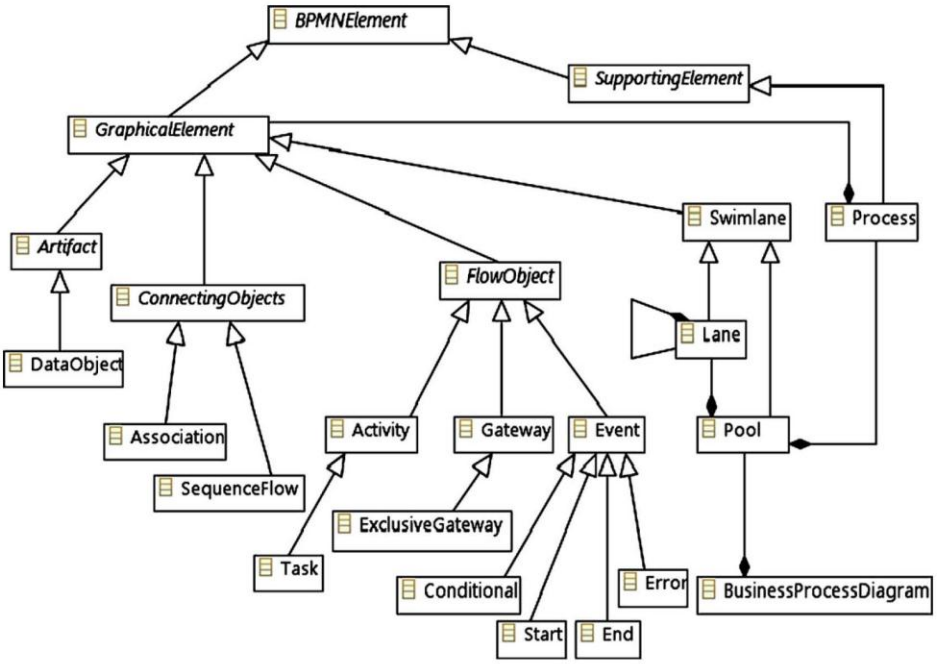


Figure 8. BPMN metamodel.

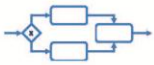
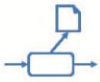
Figure 9 shows a fragment of the QVT transformation: the *Package2Pool* and *CallableUnit2Task* relations. First, the *Package2Pool* relation transforms each instance of the *Package* metaclass into a *Process* instance nested in a *Pool* instance with the same name. This transformation builds the BPD skeleton in the business process model at L3 according to the $P1$ pattern (see Table II). The *where* clause additionally triggers the *CallableUnit2Task* relation for each *CallableUnit* nested in a *CompilationUnit* that belongs to the Package instance. Second, the *CallableUnit2Task* relation transforms the *CallableUnit* instances into *Task* instances within the BPD supporting the $P2$ pattern (see Table II). In the where clause, this relation calls the *WritesStorableUnit-2DataObject* relation from among other relations, which implements the $P6$ pattern, that is called for each piece of data written by the callable unit.

After the QVT transformation, business experts can modify the recovered business process models in order to tune the organization's processes. On the one hand, this is owing to the fact that the business processes obtained may contain certain elements related to the technical nature of a legacy system, which do not depict any of the organization's business behavior. These elements can therefore be removed, or perhaps renamed. On the other hand, the preliminary business processes may lack certain business elements that depict specific aspects of the organization's business operation. They should thus be added to the recovered business processes. In this last transformation the procedure therefore defines a set of manual operations that the business experts can carry out. The finite set of operations consists of *Add* [A], *Remove* [R], *Rename* [RN], *Join* [J], and *Split* [S]. The *Join* and *Split* operations can be applied solely to whole business processes or activities, while the remaining operations can be applied to any kind of business element. In addition, the *Join* and *Split* operations can be used together with the addition of *Lanes* to represent information about several subsystems. Lanes representing subprocesses which are related by means of certain data objects written by a subprocess and read by another one (e.g. data stored in a database by a subsystem and loaded by other subsystem; or a subsystem invoking a service of another one).

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

Table II. Summary of the set of business patterns.

Sequence patterns	*Event patterns*
P1. BPD skeleton. This pattern creates the organizational structure of a business process model. It creates a business process diagram for each KDM code model and a pool element with a nested process in the diagram for each package of the KDM code model.	*P5. Start.* The task building from the callable unit that starts the execution of any program or application of the legacy system is considered as the initial task. A start event is therefore built into the BP diagram and a sequence flow from this event to the initial task is also created.
P2. Sequence. This pattern takes any callable pieces of code from the KDM code model and maps them into tasks in the BP diagram. The sequence of calls to callable units is transformed into a set of sequence flows in the tasks built from the callable unit.	*P6. Implicit termination.* This pattern builds an end event in the BP model. It then creates sequence flows from 'end task' and these flows merge in the end event. A task is considered to be an 'end task' if this task does not have any outgoing sequence flow.
 *P3. Branching.* This pattern transforms each conditional jump of the source code that has two mutually exclusive choices into an exclusive gateway and two different sequence flows in the business process model. These exclusive conditional branches are typically related to the *if-then-else* or *switch* statements in several programming languages.	*P7. Conditional sequence.* This pattern transforms each conditional call into a sequence flow triggered under a conditional intermediate event through to the task related to the callable unit. It makes it possible to create arbitrary cycles in the BP diagram.
P4. Collaboration. Each call to external callable unit (i.e. API libraries or external components outside the system like remote callable unit) is transformed into both an auxiliary task and two sequence flows in a round-trip manner.	*P8. Exception.* Each call to callable unit that manages any exception is transformed into a task and a sequence flow fired under an error intermediate event. It can be considered as a specialization of the P7 pattern.
Data patterns	
P9. Data input. This pattern transforms each piece of input data within a callable unit in the KDM code model into a data object and an association between the data object and the task previously built from the callable unit. It only considers as input data the parameters or arguments of the callable unit, but it does not consider the auxiliary variables within the callable unit.	 *P10. Data output.* Each piece of output data involved in a callable unit is transformed into a data object and an association from the task (built from the callable unit) to the data object. It excludes as output data the auxiliary and intermediate data in the body of the callable unit. The output data is the data returned by the callable unit or external data related to databases or files.

In order to reduce the subjective viewpoint of the business experts, some recommendations are provided to the business experts. These suggestions aid the decision-making process carried out by business experts. The recommendations are automatically defined by analyzing and scoring certain elements of the obtained business process models. There are mainly two kinds of recommendations. The first kind of recommendations is provided when a business process (obtained from a particular code package) has several business tasks related to other business processes (obtained from other code package by applying the pattern *P4. Collaboration*), it suggests joining both business process models obtained from two different code packages. The second kind of recommendation is provided whether a certain business process model consists of two (or more) isolated connected graphs, it suggests then that business experts split the model into two (or more) business process models.

To continue with the example, at the beginning the L2-to-L3 transformation obtains a first sketch of the BPD from the KDM model by means of the QVT transformation. Figure 10(A) shows the graphical representation of the business process model obtained during the QVT transformation.

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

```
transformation patterns (kdm:code, bpmn:bpmn){
    ...
    relation Package2Pool {
        xName : String;
        checkonly domain kdm pk : code::Package  {
            name = xName
        };

        enforce domain bpmn bpd: bpmn::BusinessProcessDiagram  {
            Pools = p : Pool {
                name = xName,
                ProcessRef = pr : bpmn::Process {
                    Name = xName
                }
            }
        };
        where {
            pk.codeElement->forAll (c:code::AbstractCodeElement |
            c.oclAsType(code::CompilationUnit).codeElement->forAll(m:code::AbstractCodeElement |
            (m.oclIsKindOf(code:: CallableUnit) and m.oclIsUndefined() and m.oclAsType(code::
            CallableUnit).name<>'main')  implies   CallableUnit2Task (m, pr)));
            ...
        }
    }
    ...
    relation CallableUnit2Task {
        xName : String;
        checkonly domain kdm m : code::CallableUnit {
            name = xName
        };
        enforce domain bpmn pr : bpmn::Process {
            GraphicalElements = t : bpmn::Task {
                Name = xName,
                Status = bpmn::StatusType::None,
            }
        };
        where {
            m.codeElement->forAll (a: AbstractCodeElement | a.oclAsType
            (ActionElement).actionRelation->forAll (w:AbstractActionRelationship |
            (w.oclIsKindOf(Writes)) and w.oclAsType(Writes).to.oclIsKindOf
            (StorableUnit) implies WritesStorableUnit2DataObject (w, t, pr)));
            ...
        }
    }
}
```

Figure 9. QVT relations to implement the L2-to-L3 transformation.

This model contains 10 tasks in total, although only four tasks are related to the four *CallableUnit* instances in the KDM model (see Figure 7), which are obtained by applying the *P2.Sequence* pattern. The gateways are also created by applying the *P3. Branching* pattern. Moreover, six other tasks are also obtained by applying the *P4.Collaboration* pattern (see Table II). This pattern is applied to the three *ActionElement* instances of the *CallableUnit* instance named *sendProducts* in the KDM model (see Figure 7). These action elements represent calls to methods not defined in the same Java source file (e.g. *update*, *setUnits*, and *insert*), and these calls are thus represented as external calls, which are transformed into six tasks with a round-trip sequence flow from the previous tasks.

Figure 10(B) shows the business process model refined by business experts. There are three principal improvements:

- The three tasks obtained from external calls are removed, since the business experts consider that these tasks do not represent any of the organization's business activities, i.e. these are related to the technical dimension of the system.

  ```
  Remove ['getCustomer];
  Remove ['getUnits'];
  Remove ['insert'];
  ```

- A gateway is added to merge the two branches opened after the first gateway.

  ```
  Add [ GATEWAY['sendProducts', 'gateway(CheckInventory)', END] ];
  ```

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

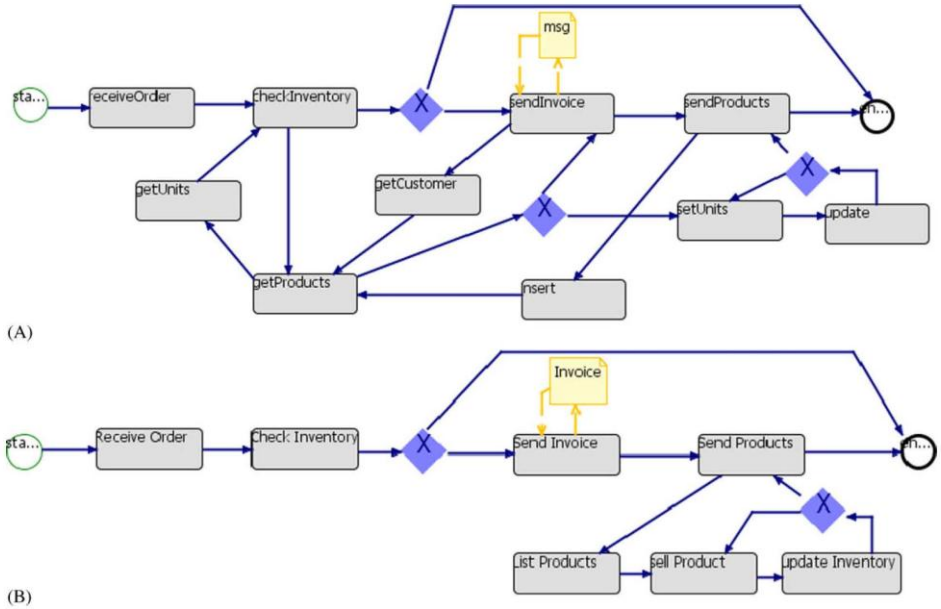


Figure 10. An example of business process models obtained for a product shipping system.

- The four remaining tasks are renamed by the business experts in order to fit the names to the organization's business activities.

```
Rename ['receiveOrder', 'Receive Order'];
Rename ['checkInventory', 'Check Inventory'];
Rename ['sendProducts', 'Send Products'];
Rename ['sendInvoice', 'Send Invoice'];
Rename ['getProducts', 'List Products'];
Rename ['update', 'Update Inventory'];
Rename ['setUnits', 'Sell Product'];
Rename ['msg', 'Invoice'];
```

Finally, Figure 10(B) shows the business process model that represents a part of the organization's behavior through the business process recovery from the piece of source code presented at the beginning of this example (see Figure 4, left-hand side).

5. A SUPPORTING TOOL

An *ad hoc* tool has been developed to semiautomate the proposed procedure. This tool facilitates the adoption of the recovery procedure as well as the execution of the case study. The tool is based on the Eclipse platform and uses four key technologies. The first technology is *JavaCC*, which is a parser generator for Java language [45]. The second technology is EMF (Eclipse Modeling Framework) [46]. This Eclipse framework makes it possible to build specific metamodels according to the *ECORE* meta-metamodel, the metamodel proposed by the Eclipse platform to define metamodels. EMF also provides tools to produce source code from *ECORE*-based metamodels to automatically build model viewers and editors. GMF (Graphical Modeling Framework) is additionally used together with EMF to generate graphical model editors. The third technology is XMI (XML Metadata Interchange), which defines the manipulation and interchange of models through XML

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

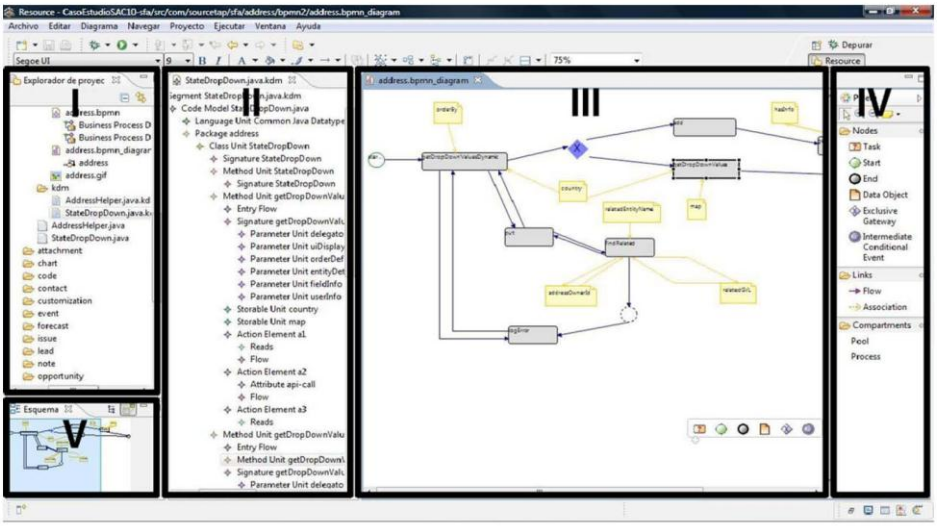


Figure 11. The tool to support the business process archeology method.

[47]. All the models involved in the procedure are thus made persistent by means of an XMI file. Finally, the fourth technology is *MediniQVT* [48], an open source framework which is a model transformation engine for QVT-Relations.

The tool architecture is divided into three different modules aligned with each of the transformations defined in the procedure. The first module supports the L0-to-L1 transformation. It thus consists of a static analyzer of Java source code, which is built though *JavaCC* from the EBNF (*Extended Backus–Naur Form*) grammar of Java 1.5. This module takes a Java file as input and then generates an XMI file as the output that represents the Java code model, a PSM model in L1 (see Figure 11(I)).

The second module executes a QVT transformation to generate the KDM model in L2 from the previous Java code models using *MediniQVT*. This module also has a tree view editor for KDM models and was built through EMF (see Figure 11(II)).

The third module also executes the QVT transformation of the L2-to-L3 transformation. Moreover, a graphical editor for BPD at L3 was developed using EMF/GMF (see Figure 11(III)), which also provide an outline view to make the work easier (see Figure 11(V)). This editor therefore allows business experts to modify and refine the business process models according to the proposed set of manual operations in order to fit the first version of business processes to the real-life organization's behavior. For this purpose, the tool provides a palette of business elements (see Figure 11(IV)).

6. CASE STUDY

This section presents a case study to validate the feasibility of the proposed business process recovery procedure by applying it to an e-government system. In order to automatically apply the proposal to the system, a tool that supports the entire depicted business process recovery procedure has been developed. The case study is carried out by following the protocol for planning, conducting and reporting case studies proposed by Brereton *et al.* [18]. The following sections present the details of the main stages defined in the formal protocol: background, design, case selection, case study procedure, data collection, analysis and interpretation, and validity evaluation.

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

Table III. Research questions of the case study.

Id	Research question
MQ	Can the recovery procedure properly obtain business processes from legacy information systems?
AQ1	Can the proposed business process recovery procedure obtain business processes with adequate accuracy levels?
AQ2	Is the proposed business process recovery procedure efficient?

6.1. Background

The previous research on this subject must first be identified. The related work presented in Section 2 shows other proposals for recovering business processes from legacy information systems, and compares these proposals with our procedure, considering four criteria. Our particular procedure is (i) a model-driven framework (more specifically, an ADM-based framework) for recovering business processes, which (ii) uses static analysis as the recovery technique, and (iii) source code as the main software artifact in addition to business expert information. Moreover, (iv) the case study is carried out by following a formal protocol for case studies [18].

The *object of study* is the proposed business process recovery procedure, and the *purpose of this study* is the evaluation of specific properties of the proposed procedure related to effectiveness and efficiency.

Bearing in mind the object and purpose of the study, the main research question (hereafter, *MQ*) addressed by the study is the following: *Can the recovery procedure properly obtain business processes from legacy systems*? In this question, the adverb *properly* means to obtain business processes in an effective and efficient way. In order to find out if *MQ* is true, Table III shows the additional research questions that are identified from the *MQ* question. The additional research question *AQ1* is established in order to evaluate whether the recovered business processes faithfully represent the organization's business behavior. That is, *AQ1* aims to evaluate the effectiveness of the procedure. Moreover, the additional research question *AQ2* (see Table III) evaluates whether the proposed business process recovery procedure obtains business processes efficiently.

6.2. Design

This case study consists of a single case, i.e. it focuses on a single legacy information system. The case study also considers several analysis units within the case. This study can therefore be considered as an *embedded* case study according to the classification proposed by Yin [49]. The analysis units are the different source code packages of the legacy information system, which is the independent variable of the study. Each source code package is represented in a model in L1 and L2 and is then transformed into a business process model in L3 through the third transformation. Despite the fact that there is a business process for each code package, the business process models obtained can be joined or split by means of the business experts' manual intervention according to the L2-to-L3 transformation.

The study consists of the analysis of each business processes recovered in order to answer the questions established in Table III. Some measures are therefore established to provide quantitative answers to the proposed research questions. First, in order to evaluate the effectiveness of the proposed procedure through the *AQ1* question, the study proposes the use of two measures: *Precision* and *Recall*. These measures were designed for information retrieval scenarios, although they are usually applied to other mining scenarios like business process recovery among others [50].

On the one hand, the *Precision* measure indicates the amount of relevant recovered elements within the set of recovered elements in a business process model. An element is considered relevant if this element faithfully represents the organization's business operations or business behavior in the real world. On the other hand, the *Recall* measure represents the amount of relevant recovered elements of the total of relevant elements (recovered and not recovered), which depicts the organization's entire business operation. While source code package is the independent variable, these two measures are dependent variables in the study.

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

The study considers the *task* element as the score unit in order to apply these measures in a business process recovery scenario. The *Precision* measure (1) is therefore defined as the number of true relevant tasks divided by the total number of relevant (recovered) tasks, i.e. the sum of true relevant tasks and false relevant tasks that were incorrectly recovered. Moreover, the *Recall* measure (2) is defined as the number of true relevant tasks divided by the total number of relevant tasks, i.e. the relevant tasks of the business process and other tasks that should have been recovered but were not recovered. Although *Precision* and *Recall* measures are adequate, there is an inverse relationship between them. The extraction of conclusions with an isolated evaluation of these metrics is consequently very difficult. These measures are therefore usually combined into a single measure known as an *F-measure* (3), which consists of a weighted harmonic mean of both measures.

$$PRECISION = \frac{|\{relevant\ tasks\} \cap \{recovered\ tasks\}|}{|\{recovered\ tasks\}|} \tag{1}$$

$$RECALL = \frac{|\{relevant\ tasks\} \cap \{recovered\ tasks\}|}{|\{relevant\ tasks\}|} \tag{2}$$

$$Fmeasure = \frac{2 \cdot PRECISION \cdot RECALL}{PRECISION + RECALL} \tag{3}$$

We also use the business expert opinion to discover which recovered tasks are relevant or not in order to evaluate the proposed measures. In this respect, the definition of *relevant task* is a key factor in facilitating the business experts' work, and it must be defined *a priori*. Despite the fact that the evaluation of measures focuses only on the business tasks, the *relevant task* definition implicitly considers other business elements, since the execution flow must not be ignored. The *relevant task* definition establishes a set of four conditions. Condition $C1$ specifies that the task must represent a real-life business operation within the organization. For instance, the task named '*parseDate*' obtained from an auxiliary Java method does not represent any valuable business activity. This condition must not be evaluated by considering task names, since these names are inherited from legacy code and they may be biased as regards the real business activity names provided by business experts. For example, the task named *receiveOrder*, despite its name is not exactly *Receive Product Orders*, it represents the valuable business activity in charge of receiving product orders. Condition C2 ensures that all the recovered tasks preceding the evaluated task must be relevant tasks. In order to meet C2, there can be no non-relevant tasks with a sequence flow to the evaluated task. In a similar way, Condition C3 ensures that all the following tasks must be directly (or indirectly) recovered relevant tasks. The conditions C2 and C3 check the control flow of business process models focusing on recovered tasks one by one. Finally, Condition C4 specifies that all the *Data Object* elements read and written by the evaluated task must also be recovered.

Second, in order to answer question *AQ2*, this study evaluates the total time spent on executing each transformation. The transformation time values are automatically measured for each analysis unit of the study by the tool developed. The time values obtained are analyzed with regard to the total number of elements built into each specific business process model. The purpose is to discover if the proposed recovery procedure can be scalable to a larger legacy system. Both the size of the business process models and the time values are also considered as the dependent variables of this study.

6.3. Case selection

The case selection is a key stage in the case study planning, which aims to select a good and suitable case to be studied. Table IV presents the five criteria established to select the most appropriate software system. *Cr1* guarantees that the legacy system selected is an information system that supports an organization or company's business operation. This criterion discards, for example, embedded systems or real-time systems. *Cr2* ensures that the legacy system will be a real-life

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

Table IV. Criteria for case selection.

Id	Criteria for case selection
Cr1	It must be an enterprise system
Cr2	It must be a real-life system
Cr3	It must be a legacy system
Cr4	It must be of a size not less than 100 KLOC
Cr5	It must be a Java-based system

system that is deployed in a production environment. *Cr3* ensures that the selected system really is a legacy information system. To evaluate this criterion the time in production is not used, since it is not a good measure. Instead of production time, *Cr3* considers the amount of modifications in the system that alter the business processes, i.e. the system modifications related to the *adaptive* and *perfective* maintenance according to [51]. *Cr4* ensures that the system is not a *toy program*, since it defines a threshold of 100 000 lines of source code. Finally, *Cr5* guarantees that the system is based on the Java platform, since both the proposed recovery procedure and the tool were developed for Java-based systems.

After evaluating several available systems according to the previous criteria, the legacy information system that was selected to be studied was *eAdmin*, an e-government system. This system supports the electronic administration of a Spanish regional ministry of Housing and Urban Development. This system automates all the services offered by the government and their document management, thus meeting the *Cr1* criterion. The first release of the system was moved to the production stage 26 months ago, thus meeting the *Cr2* criterion. During this time, the same development team of the *eAdmin* system has made three medium modifications (versions 1.1, 1.2, and 1.3), a large modification (version 2.0), and two medium modifications (versions 2.1 and 2.2). This ensures compliance with the *Cr3* criterion. From a technological point of view, *eAdmin* is a Web application and its architecture is separated into three layers: presentation, business, and persistence. The technology used to develop the presentation layer was *JSP* (*Java Server Pages*), *Java* for business layer, and *Oracle* together with *JDBD-ODBC* for the persistence layer. The total size of the legacy system is 320.2 KLOC. Criteria *Cr4* and *Cr5* are therefore also met.

6.4. Case study procedure

In addition to the design and case selection of the case study, the execution procedure of the study must also be planned. The execution is aided by the tool developed to support the procedure. The case study procedure defines the followings steps:

1. After some meetings between the staff of the candidate organizations and researchers, a legacy information system is selected according to the case selection criteria. The business expert needed to perform the manual post verification is also selected in this step. In this study, this person is the *chief information officer* of the Spanish regional ministry of Housing and Urban Development, since this person has the highest level of expertise with regard to the organization's business activities.
2. The legacy system under study is implanted in an experimental environment: the source code is deployed in a Web server, the database schema is built through the database scripts, the initial data is loaded, etc.
3. The different business process models are obtained from the legacy source code using the tool that supports the procedure (cf. Section 5). The execution hardware environment consists of a computer with a RAM memory of 4 GB and two processors each of 2.67 GHz.
4. The first sketch of business processes obtained through the model transformation is improved by business experts. They fit the preliminary business processes with the reality of the organization, i.e. they can add tasks that should have been recovered but were not recovered, or remove tasks erroneously recovered. After business expert intervention, the accuracy of business process models is evaluated by comparing each preliminary business process and its

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

related business process enhanced by business experts. We obtain the value of the proposed measures like precision and recall by scoring the differences between the preliminary and enhanced business processes.

5. The key information related to the generation of business processes (cf. step 3), along with the business expert intervention (cf. step 4), is collected for each analysis unit according to the data collection plan (cf. Section 6.5).
6. The data collected in the previous step are analyzed and interpreted to draw conclusions in order to answer the research questions. Section 6.6 shows the outgoing results obtained in this case study. Finally, the case study is reported and the feedback is given to the organization and research community.

6.5. Data collection

The data to be collected and the data sources must be defined before starting the execution of the case study. The following information is thus recorded for each analysis unit (see Table V): (i) the number of source code files in each package; (ii) the number of business process models obtained after the L2-to-L3 transformation; (iii) the total number of elements in the business process model (the size of the model); (iv) the total number of recovered tasks in order to evaluate after the Precision and Recall metrics; (v) the total transformation time in milliseconds. Finally, the bottom part of Table V summarizes all the information by means of the total, mean and standard deviation for each data column.

The right-hand side of Table V also summarizes the manual interventions by the business expert, and thus presents (i) the ID number of each preliminary business process model and (ii) the inter-model operation(s) carried out in each model or between models (e.g. rename, remove, join and split models).

Table VI shows (i) the number of tasks recovered (before manual intervention); (ii) the number of recovered relevant tasks, i.e. the number of tasks that the business expert marked as correct; (iii) the number of recovered non-relevant tasks, i.e. the tasks removed from the business process since they did not represent a business activity; (iv) *Precision* and (v) *Recall* values for each final business process; and (vi) the harmonic mean between them. The measures are calculated with data from the previous columns of this table. Table VI also presents (i) the total number of business elements in each final business process and (ii) the total transformation times, which are derived through aggregation from the data collected in Table V.

6.6. Analysis and interpretation

After the data have been collected, they are analyzed in order to draw conclusions. The analysis should obtain the evidence chains from data to answer the research questions. In order to answer the question *AQ1*, Figure 12 shows the box chart for *Precision* and *Recall* measures. The chart shows the density distributions for the set of final business processes. The mean of the distribution of the *Precision* measure (0.66) is lower than the mean of *Recall* measure (0.91). Moreover, the standard deviation of the *Recall* measure is close to 0 (0.05), i.e. the central values of the *Recall* distribution are more concentrated around the mean than the central values of the *Precision* measure (0.13).

The interpretation of these values is the following. On the one hand, a higher *Recall* value means that the proposed procedure recovers a higher number of relevant business elements, i.e. it recovers most of the tasks from the current business processes. On the other hand, the *Recall* value contrasts with the low *Precision* value, which means that the number of non-relevant tasks is very high with regard to the recovered tasks. A low *Precision* and a high *Recall* mean that the preliminary business processes were obtained as large business processes with almost all the relevant tasks (high *Recall* value), but they were obtained with a great amount of non-relevant tasks (low *Precision* value) that were removed by business experts. The low *Precision* value, i.e. the significant amount of non-relevant tasks, is due to the fact that some tasks are obtained almost directly from the source code and they are related to the technical nature, and do not therefore represent any piece of the embedded business knowledge.

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

Table V. Data collected in the case study.

Package	# source code files	# BPMN models	# Elements	# Tasks	Transf. time (ms)	Preliminary BP Id	Manual intervention in business process models
d.seventhLaw	32	1	347	73	2001	1	J[22, 23], RN ['Administration']
d.admin	75	1	485	77	5032	2	R
d.spfPurchase	44	1	556	70	5134	3	J[24], RN['Social Protection Floor MGMT']
d.enclose	11	1	344	27	2548	4	J[25], RN['Document MGMT']
d.rent	140	1	1855	265	22 850	5	J[26],RN['Rental House MGMT']
d.rnDocument	104	1	922	189	10 195	6	J[28], RN['Renovation Document Registration']
d.common	82	1	604	121	5416	7	R
d.applicant	53	1	1340	191	12 804	8	J[29], RN['House Applicant MGMT']
d.ppFiles	57	1	622	117	4690	9	J[30], RN['Personal File MGMT']
d.ruralHouse	38	1	595	107	8427	10	J[31], RN['Rural House MGMT']
d.developer	64	1	795	135	6348	11	J[32],RN['Developer MGMT']
d.renovation	20	1	511	54	10 037	12	R
d.rnFacade	28	1	783	113	8666	13	J[34], RN['Renovation MGMT']
d.grants	39	1	471	85	4325	14	J[35], RN['Emancipation Grant MGMT']
d.secondHand	38	1	476	74	3369	15	J[37], RN['Second-Hand House MGMT']
d.user	10	1	216	48	1040	16	R
pdfhandler	6	1	490	55	2364	17	R
eRegistration	8	1	82	14	4385	18	R
security	11	1	145	30	1077	19	R
util	25	1	340	92	1795	20	R
web	9	1	322	67	1518	21	R
w.seventhLaw	10	1	76	12	503	22	J[1,23], RN['Administration']
w.admin	88	1	313	49	2267	23	J[1,22], RN['Administration']
w.spfPurchase	9	1	143	27	761	24	J[3], RN['Social Protection Floor MGMT']
w.enclose	6	1	15	6	176	25	J[4], RN['Document MGMT']
w.rent	29	1	126	25	533	26	J[5],RN['Rental House MGMT']
w.help	1	1	69	1	86	27	R
w.rnDocument	27	1	125	15	543	28	J[6],RN['Renovation Document Registration']
w.applicant	15	1	154	22	406	29	J[8],RN['House Applicant MGMT']
w.ppFiles	20	1	112	17	559	30	J[9],RN['Personal File MGMT']
w.ruralHouse	7	1	79	11	1956	31	J[10],RN['Rural House MGMT']
w.developers	22	1	29	11	686	32	J[11],RN['Developer MGMT']
w.renovation	3	1	16	4	186	33	R

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

Table V. *Continued.*

Package	# source code files	# BPMN models	# Elements	# Tasks	Transf. time (ms)	Preliminary BP Id	Manual intervention in business process models
w.rnFacade	12	1	143	19	1420	34	J[13],RN['Renovation MGMT']
w.grants	12	1	196	37	976	35	J[14],RN['Emancipation Grant MGMT']
w.session	2	1	1	0	110	36	R
w.secondHand	11	1	181	31	933	37	J[15],RN['Second-Hand House MGMT']
w.user	13	1	88	19	560	38	R
w.validator	2	1	12	2	109	39	R
Total	1183	39	14179	2312	136791		
Mean	30.3	1.0	363.6	59.3	3507.5		
Std. deviation	31.6	0.0	384.3	59.9	4556.9		

The results obtained are usual since there is an inverse relationship between *Precision* and *Recall* measures. Figure 13 represents the inverse relationship between both measures, and shows the results obtained (high recall, but low precision). The *Precision* value should, ideally, always be 1.0 for any *Recall* value but, according to [52], this is not possible in practice (see Figure 13). Thus, owing to the fact that the relationship between both measures has an inverse nature, it is possible to increase one at the cost of reducing the other. Indeed, the proposed procedure could reduce its *Recall* value by recovering fewer tasks, at the cost of reducing the number of non-relevant recovered tasks, i.e. increasing the *Precision* value. This hypothetical result is more desirable than the obtained result, since the *Precision* and *Recall* values would be more balanced.

Indeed, the *F-measure* (3) can be considered as a special case of the general F_α*-measure* (4), where α is 1. The α value is used to weight the *Precision* in relation to *Recall*, and the selected F_1*-measure* (3) thus signifies that both *precision* and *recall* are equally important. A good balance between *Precision* and *Recall* would therefore be better to obtain a higher F_1*-measure* value

$$F_\alpha = \frac{(1+\alpha)\cdot PRECISION \cdot RECALL}{\alpha \cdot PRECISION + RECALL} \tag{4}$$

In order to jointly evaluate both measures, Figure 14 shows the F_1*-measure* values obtained, along with the *Precision* and *Recall* values for each final business process. The F_1*-measure* has a mean of 0.76 and a standard deviation of 0.10.

These values are additionally compared with reference values from other experiences with model recovery in the literature, such as those of [53–55]. We found reports of *Precision* and *Recall* values close to 60%, and these were our benchmark values. The values obtained for our measures ($Precision = 66\%$, $Recall = 91\%$ and $F_1 - measure = 76\%$) were therefore clearly above 60%, the reference value. Question *AQ1* can consequently be positively answered, i.e. the proposed procedure makes it possible to recover business processes from legacy information systems with adequate accuracy levels. However, the *F-measure* value might be slightly improved by obtaining more balanced *Precision* and *Recall* values.

The transformation time was also analyzed in order to answer question *AQ2*. After applying the proposed recovery procedure, 39 preliminary business process models were obtained with an average size of 363.6 elements (or 59.3 tasks) per BPD (see Table VI). The total time was 137 s (2'28" approximately), and the processing rate is thus equal to 2.65 elements per second. Finally, the time spent on processing each task was 2.31 s.

Despite the fact that the mean of the *Precision* is a medium value in this study, the time wasted as a result of recovering non-relevant tasks is low. The total time if we consider only the final business process models was, therefore, 103 s, and the time spent on recovering non-relevant processes thus represents only 24.8%.

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

Table VI. Data of final business process models.

Business process model	# Recovered Tasks (RcT)	# Recovered relevant tasks (RcRvT)	# Recovered non-relevant task (RcNRvT)	# Non-recovered relevant tasks (NRcRvT)	Precision (RcRvT/RcRvT+ +RcNRvT)	Recall (RcRvT/RcRvT+ NRcRvT)	F-Measure (harmonic) (mean)	# Elements	Transf. time (ms)
Administration	134	69	65	10	0.51	0.87	0.65	736	4771
Social Protection Floor MGMT	97	64	33	6	0.66	0.91	0.77	699	5895
Document MGMT	33	23	10	3	0.70	0.88	0.78	359	2724
Rental House MGMT	290	221	69	9	0.76	0.96	0.85	1981	23 383
Renovation Document Reg.	204	169	35	3	0.83	0.98	0.90	1047	10 738
House Applicant MGMT	213	190	23	9	0.89	0.95	0.92	1494	13 210
Personal File MGMT	134	74	60	9	0.55	0.89	0.68	734	5249
Rural House MGMT	118	63	55	12	0.53	0.84	0.65	674	10 383
Developer MGMT	146	87	59	15	0.60	0.85	0.70	824	7034
Renovation MGMT	132	91	41	3	0.69	0.97	0.81	926	10 086
Emancipation Grant MGMT	122	62	60	7	0.51	0.90	0.65	667	5301
Second-Hand House MGMT	105	76	29	9	0.72	0.89	0.80	657	4302
Mean	144	99.08	44.92	7.92	0.66	0.91	0.76	899.8	8589.7
Std. deviation	65.5	60.30	18.94	3.73	0.13	0.05	0.10	437.1	5635.0

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

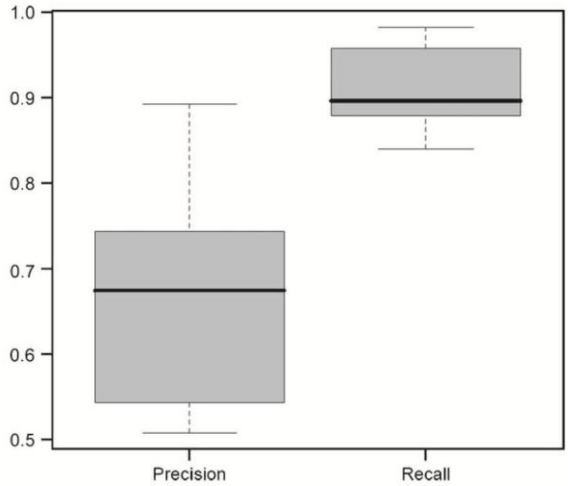


Figure 12. The box chart for *Precision* and *Recall* measures.

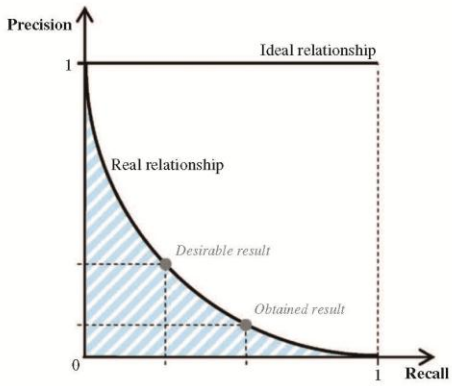


Figure 13. Relationship between *Precision* and *Recall* measures, and obtained results.

A total time of 137 s for the proposed transformations seems feasible for the selected case, since the size of the *eAdmin* is 320.2 *KLOC*. Nevertheless, the scalability of the procedure must be evaluated. Under the hypothesis that the time complexity of the procedure theoretically is linear, we established a linear regression model to check it and find out if the proposal is scalable. The linear regression model considers the transformation time as a dependent variable and the size of the analysis units as an independent variable. Figure 15 shows the scatter chart of size/time, which represents two charts considering (a) the total number of elements (circle points) and (b) the number of tasks (square points) as the size of each business process model. This chart also shows the regression lines, which in both cases present a positive linear relationship between the model size and the time spent on model transformations.

Moreover, Figure 15 shows the correlation coefficient R^2, which is the degree to which the real values of the dependent variable are close to the predicted values, i.e. how much the points are fitted to the regression line. In this study, the correlation coefficient was 0.89, considering the total elements as the model size, and 0.79 considering the task number as size. Since the correlation coefficient value is between 0 and 1 for a positive linear relationship, these values are very high. This signifies that the proposed linear regression model is suitable to explain the data obtained in

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

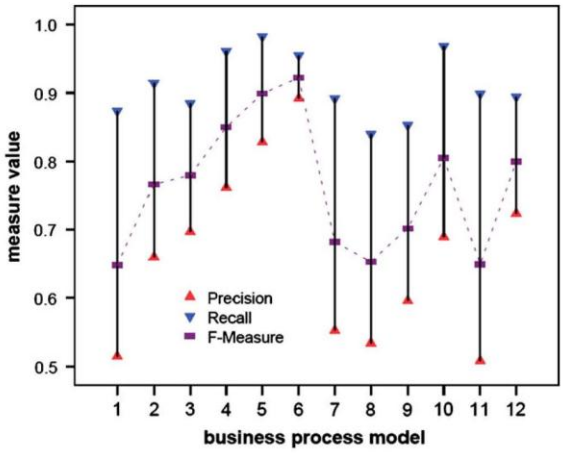


Figure 14. F-Measure values for the final business processes.

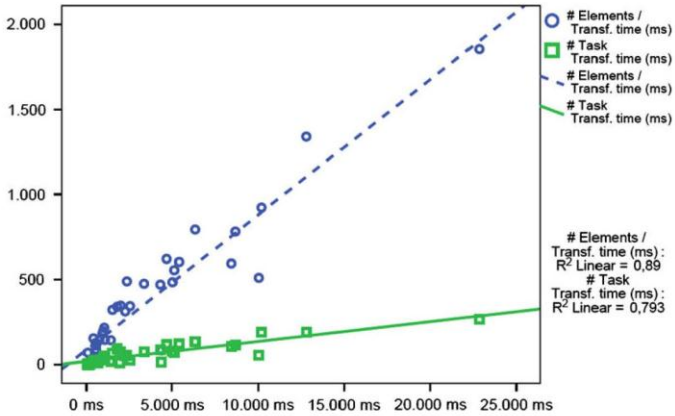


Figure 15. The size/time scatter chart.

this study, i.e. there is no quadratic or exponential relationship between transformation time and model size. The increase in time for larger systems will consequently be linear, and the time will thus be assumable.

This procedure is semiautomatic, thus the manual effort carried out by business experts must also be analyzed besides the machine computation time. The selected business expert took 96.3 min to refine the first version of the business processes obtained by the tool. That effort focused on the refinement of 39 preliminary business process models into 12 final models and the refinement implied a total reduction concerning tasks over 37%. As a consequence, the total time spent on recovering the embedded business processes through the proposed procedure was 98.8 min. The manual time spent on this study is greater than the computational time. However, the manual time related to our proposal would be less than the time spent on a manual business process redesign from scratch. In addition, the manual time will probably be linear with respect to the size of legacy information systems in the same way as the computational time.

In conclusion, the *AQ2* question can therefore be answered as true, and the main research question *MQ* is also answered as true, i.e. the proposed business process recovery procedure makes

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

it possible to obtain business process models from legacy information systems in an effective and efficient way.

6.7. *Validity evaluation*

Finally, the validity of the results obtained in the case study must be evaluated. This stage evaluates whether the results are true and not biased for the whole population to which we would generalize the results. This section thus shows the threats to the validity of this case study. According to [56] there are three main types of validities: internal, construct, and external validity.

6.7.1. Internal validity. There is no large population that makes it possible to obtain statistically representative results, although a clear trend for the proposed measures was identifiable in this case study. However, there are two determining factors related to obtaining the results presented here. First, the tool used to obtain the business processes could be a factor that affects the transformation time. This means that the measure values might be different if the business processes were obtained with another tool supporting the same recovery procedure. Second, if this study is replicated with other cases involving a different business expert, the results concerning *Precision* and *Recall* measures might have some deviations as a result of the business expert's manual intervention, since this person provides his/her subjective point of view.

6.7.2. Construct validity. The measures of the case study were adequate to measure the variables and answer the research questions appropriately. The *Precision* and *Recall* measures were reused from the information retrieval field, in which these metrics have an adequate maturity level. These measures also allow us check whether or not the business processes obtained faithfully represent the organization's business operative, since these values can be compared with reference values. Moreover, the time and size measures allow us to answer the research question regarding the scalability of the proposal.

6.7.3. External validity. Finally, *external validity* is concerned with the generalization of the results. This study considers traditional legacy information systems as the whole population. In this respect, the results obtained could be generalized to this population. However, the specific platform of the selected case is a threat that should be noted. The results are thus strictly extended to those legacy information systems based on object-oriented languages like *Java*.

7. CONCLUSIONS AND FUTURE WORK

This paper has addressed the problem of business process recovery from legacy systems in order to preserve valuable business knowledge. We have therefore presented a recovery procedure as a possible solution, which is framed in MARBLE (an ADM-based framework). The proposed procedure is characterized by three main features: (i) it focuses on legacy source code as the main source of knowledge, (ii) it uses static analysis as a reverse engineering technique to extract the information needed; and (iii) it follows the model-driven development principles, i.e. it considers different models at different abstraction levels and a set of model transformations between them. Furthermore, we have also developed a Java supporting tool in order to automate the procedure and provide mechanisms for its adoption.

In order to validate the proposal, we have presented a real-life case study as a key contribution of this paper. The selected case is an e-government system based on Java, which is used in the application of the proposed recovery procedure. The case study aims to evaluate both the effectiveness and efficiency of the proposal. In order to evaluate the effectiveness, we use the *Precision* and *Recall* measures. We believe that these metrics are adequate to estimate the degree to which the recovered business processes faithfully represent the organization's business operation. Moreover, in order to evaluate the efficiency, the case study evaluates the recovery time with

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

regard to the size of each business model. This evaluation allows us to check the scalability of our proposal to large legacy systems.

The results obtained in the study show that the proposed procedure is suitable for recovering business processes from legacy information systems from the point of view of effectiveness and efficiency. That is, the procedure is able to obtain business processes that depict an organization's business behavior. Moreover, the case study was planned, conducted, and reported by following a formal protocol for conducting case studies. This study can consequently be adequately replicated in the future using other legacy information systems.

Our future work will address the identified threats to the validity. On the one hand, we hope to use meta-analysis to contrast the results of this case with the results obtained for the same study concerning other legacy information systems. On the other hand, we hope to replicate this study with legacy systems based on other platforms or languages in order to compare and generalize (if possible) the results obtained.

In addition to improving the validation of the proposed procedure, we also hope to refine it. In this respect, we aim to incorporate the dynamic analysis in order to extract more and valuable business knowledge. Other software artifacts could also be considered as databases or user interfaces in the future.

ACKNOWLEDGEMENTS

This work was supported by the FPU Spanish Program; by the R+D projects funded by JCCM: ALTAMIRA (PII2I09-0106-2463), INGENIO (PAC08-0154-9262) and PRALIN (PAC08-0121-1374), and the PEGASO/MAGO project (TIN2009-13718-C02-01) funded by MICINN and FEDER.

REFERENCES

1. Weske M. *Business Process Management: Concepts, Languages, Architectures*. Springer: Berlin/Heidelberg, Leipzig, Alemania, 2007; 368.
2. Jeston J, Nelis J, Davenport T. *Business Process Management: Practical Guidelines to Successful Implementations*, (2nd edn). Butterworth-Heinemann (Elsevier Ltd.): NV, U.S.A., 2008; 469.
3. Heuvel W-JVD. *Aligning Modern Business Processes and Legacy Systems: A Component-Based Perspective* (*Cooperative Information Systems*). The MIT Press: Cambridge, MA, 2006.
4. Visaggio G. Ageing of a data-intensive legacy system: Symptoms and remedies. *Journal of Software Maintenance* 2001; **13**(5):281–308.
5. Lehman MM. Program evolution. *Information Processing and Management* 1984; **20**(1–2):19–36.
6. Koskinen J, Ahonen J, Lintinen H, Sivula H, Tilus T. Estimation of the business value of software modernizations. Information Technology Research Institute, University of Jyväskylä, 2004.
7. Paradauskas B, Laurikaitis A. Business knowledge extraction from legacy information systems. *Information Technology and Control* 2006; **35**(3):214–221.
8. Canfora G, Di Penta M. New frontiers of reverse engineering. *Future of Software Engineering*. IEEE Computer Society: Silver Spring, MD, 2007.
9. Chikofsky EJ, Cross JH. Reverse engineering and design recovery: A taxonomy. *IEEE Software* 1990; **7**(1):13–17.
10. Khusidman V, Ulrich W. Architecture-driven modernization: Transforming the enterprise. DRAFT V.5, OMG, 2007; 7. Available at: http://www.omg.org/docs/admtf/07-12-01.pdf [12 October 2007].
11. Sneed HM. Estimating the costs of a reengineering project. *Proceedings of the 12th Working Conference on Reverse Engineering*. IEEE Computer Society: Silver Spring, MD, 2005; 111–119.
12. OMG. Architecture-driven modernization standards roadmap, 2009. Available at: http://www.omg.org/docs/admtf/07-12-01.pdf [29 October 2009].
13. Newcomb P. Architecture-driven modernization (ADM). *Proceedings of the 12th Working Conference on Reverse Engineering*. IEEE Computer Society: Silver Spring, MD, 2005.
14. Pérez-Castillo R, García-Rodríguez de Guzmán I, Ávila-García O, Piattini M. MARBLE: A modernization approach for recovering business processes from legacy systems. *International Workshop on Reverse Engineering Models from Software Artifacts* (*REM'09*). Simula Research Laboratory Reports: Lille, France, 2009; 17–20.
15. Lewis GA, Smith DB, Kontogiannis K. A research agenda for service-oriented architecture (SOA): Maintenance and evolution of service-oriented systems. Software Engineering Institute, 2010; 40.
16. ISO/IEC, ISO/IEC DIS 19506. Knowledge discovery meta-model (KDM), v1.1 (Architecture-Driven Modernization), 2009; 302. Available at: http://www.iso.org/iso/catalogue_detail.htm?csnumber=32625 [19 March 2009].

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

17. Pérez-Castillo R, García-Rodríguez de Guzmán I, Piattini M. Implementing business process recovery patterns through QVT transformations. *International Conference on Model Transformation* (*ICMT'10*). Springer: Málaga, Spain, 2010; 168–183.
18. Brereton P, Kitchenham B, Budgen D, Li Z. Using a protocol template for case study planning. *Evaluation and Assessment in Software Engineering* (*EASE'08*), Bari, Italia, 2008; 1–8.
19. Daga A, Cesare SD, Lycett M, Partridge C. An ontological approach for recovering legacy business content. *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences* (*HICSS'05*), *Track 8*, *vol. 8*. IEEE Computer Society: Silver Spring, MD, 2005; 224.1.
20. OMG, Why do we need standards for the modernization of existing systems? (OMG ADM Task Force), 2003.
21. Kazman R, Woods SG, Carrière SJ. Requirements for integrating software architecture and reengineering models: CORUM II. *Proceedings of the Working Conference on Reverse Engineering* (*WCRE'98*). IEEE Computer Society: Silver Spring, MD, 1998.
22. Miller J, Mukerji J. MDA Guide Version 1.0.1. OMG, 2003; 62. Available at: www.omg.org/docs/omg/03-06-01.pdf [21 April 2010].
23. OMG. QVT. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, 2008. Available at: http://www.omg.org/spec/QVT/1.0/PDF [21 March 2009].
24. OMG. Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM), v1.1, 2009; 308. Available at: http://www.omg.org/spec/KDM/1.1/PDF/ [2 January 2009].
25. Moyer B. Software Archeology. Modernizing Old Systems. Embedded Technology Journal, 2009. Available at: http://adm.omg.org/docs/Software_Archeology_4-Mar-2009.pdf [1 June 2009].
26. OMG. Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM), v1.0, 2008; 310. Available at: http://www.omg.org/docs/formal/08-01-01.pdf [7 August 2008].
27. Cai Z, Yang X, Wang W. Business process recovery for system maintenance—An empirical approach. *25th International Conference on Software Maintenance* (*ICSM'09*). IEEE CS: Edmonton, Canada, 2009; 399–402.
28. Eisenbarth T, Koschke R, Simon D. Locating features in source code. *IEEE Transactions on Software Engineering* 2003; **29**(3):210–224.
29. Wang X, Sun J, Yang X, He Z, Maddineni S. Business rules extraction from large legacy systems. *Proceedings of the Eighth Euromicro Working Conference on Software Maintenance and Reengineering* (*CSMR'04*). IEEE Computer Society: Silver Spring, MD, 2004.
30. Zou Y, Lau TC, Kontogiannis K, Tong T, McKegney R. Model-driven business process recovery. *Proceedings of the 11th Working Conference on Reverse Engineering* (*WCRE 2004*). IEEE Computer Society: Silver Spring, MD, 2004; 224–233.
31. Ghose A, Koliadis G, Chueng A. Process discovery from model and text artefacts. *IEEE Congress on Services* (*Services'07*), Salt Lake City, UT, 2007; 167–174.
32. Paradauskas B, Laurikaitis A. Business knowledge extraction from legacy information systems. *Journal of Information Technology and Control* 2006; **35**(3):214–221.
33. Pérez-Castillo R, García-Rodríguez de Guzmán I, Caballero I, Polo M, Piattini M. PRECISO: A reengineering process and a tool for database modernisation through web services. *24th Annual ACM Symposium on Applied Computing* (*SAC'09*), Hawaii, U.S.A., 2009; 2126–2133.
34. Di Francescomarino C, Marchetto A, Tonella P. Reverse engineering of business processes exposed as web applications. *13th European Conference on Software Maintenance and Reengineering* (*CSMR'09*). IEEE Computer Society: Fraunhofer IESE, Kaiserslautern, Germany, 2009; 139–148.
35. Günther CW, van der Aalst WMP. A generic import framework for process event logs. *Business Process Intelligence Workshop* (*BPI '06*) (Lecture Notes in Computer Science, vol. 4103). Springer: Berlin, 2007; 81–92.
36. Ingvaldsen JE, Gulla JA. Preprocessing support for large scale process mining of SAP transactions. *Business Process Intelligence Workshop* (*BPI'07*) (Lecture Notes in Computer Science, vol. 4928). Springer: Berlin; 2008; 30–41.
37. van der Aalst W, Reijers H, Weijters A. Business process mining: An industrial application. *Information Systems* 2007; **32**(5):713–732.
38. Cornelissen B, Zaidman A, Deursen AV, Moonen L, Koschke R. A systematic survey of program comprehension through dynamic analysis. *IEEE Transactions on Software Engineering* 2009; **35**(5):684–702.
39. OMG. Business Process Model and Notation (BPMN) 2.0. Object Management Group, 2009; 496.
40. Müller HA, Jahnke JH, Smith DB, Storey M-A, Tilley SR, Wong K. Reverse engineering: A roadmap. *Proceedings of the Conference on the Future of Software Engineering*. ACM: Limerick, Ireland, 2000.
41. Pérez-Castillo R, García-Rodríguez de Guzmán I, Ávila-García O, Piattini M. Business process patterns for software archeology. *25th Annual ACM Symposium on Applied Computing* (*SAC'10*). ACM: Sierre, Switzerland, 2010; 165–166.
42. Aalst WMPVD, Hofstede AHMT, Kiepuszewski B, Barros AP. Workflow patterns. *Distributed and Parallel Databases* 2003; **14**(3):5–51.
43. Zdun U, Hentrich C, Dustdar S. Modeling process-driven and service-oriented architectures using patterns and pattern primitives. *ACM Transactions on the Web* 2007; **1**(3):14.
44. Zhao L, Macaulay L, Adams J, Verschueren P. A pattern language for designing e-business architecture. *Journal of Systems and Software* 2008; **81**(8):1272–1287.
45. Open Source Initiative, JavaCC 4.2. A parser/scanner generator for java, 2009. Available at: https://javacc.dev.java.net/ [3 February 2010].

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

46. EMF, Eclipse Modeling Framework Project. The Eclipse Foundation. IBM Corporation, 2009. Available at: http://www.eclipse.org/modeling/emf/ [15 December 2009].
47. OMG. XML Metadata Interchange. MOF 2.0/XMI Mapping, v2.1.1. OMG, 2007. Available at: http://www.omg.org/spec/XMI/2.1.1/PDF [7 May 2007].
48. ikv++, Medini QVT. ikv++ technologies ag, 2008. Available at: http://www.ikv.de/index.php?option=com_content&task=view&id=75&Itemid=77 [25 October 2010].
49. Yin RK. Case study research. *Design and Methods* (3rd edn). Sage: London, 2003.
50. Pradel M, Bichsel P, Gross TR. A framework for the evaluation of specification miners based on finite state machines. *26th IEEE International Conference on Software Maintenance* (*ICSM'10*), Timioara, Romania, 2010.
51. ISO/IEC, ISO/IEC 14764:2006. Software Engineering—Software Life Cycle Processes—Maintenance, 2006. Available at: http://www.iso.org/iso/catalogue_detail.htm?csnumber=39064 [23 August 2006].
52. Davis J, Goadrich M. The relationship between Precision-Recall and ROC curves. *Proceedings of the 23rd International Conference on Machine Learning*. ACM: Pittsburgh, PA, 2006; 233–240.
53. Lucrédio D, Fortes RPM, Whittle J. MOOGLE: A model search engine. *11th International Conference on Model Driven Engineering Languages and Systems*. Springer: Toulouse, France, 2008; 296–310.
54. Ye Y, Fischer G. Supporting reuse by delivering task-relevant and personalized information. *24th International Conference on Software Engineering*. ACM: Orlando, FL, 2002; 513–523.
55. Garcia VC, Lucrédio D, Durão FA, Santos ECR, Almeida ESD, Fortes RPDM, Meira SRDL. From specification to experimentation: A software component search engine architecture. *Ninth International Symposium on Component-Based Software Engineering* (*CBSE 2006*). Springer: Västerås, Sweden, 2006; 82–97.
56. Wohlin C, Runeson P, Höst M, Magnus OC, Regnell B, Wesslén A. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers: Dordrecht, 2000; 204.

Copyright © 2011 John Wiley & Sons, Ltd.

DOI: 10.1002/spe

3.3. Dynamic Approach

This section presents a dynamic technique framed in MARBLE, which obtain event logs from the execution of instrumented versions of legacy information systems (see Figure 11). A legacy information system is first instrumented by injecting a tracing statement in particular callable units of the source code. The modified version of the systems is then executed and events are recorded in logs. Finally, event logs are analyzed by process mining algorithms to discover business process models.

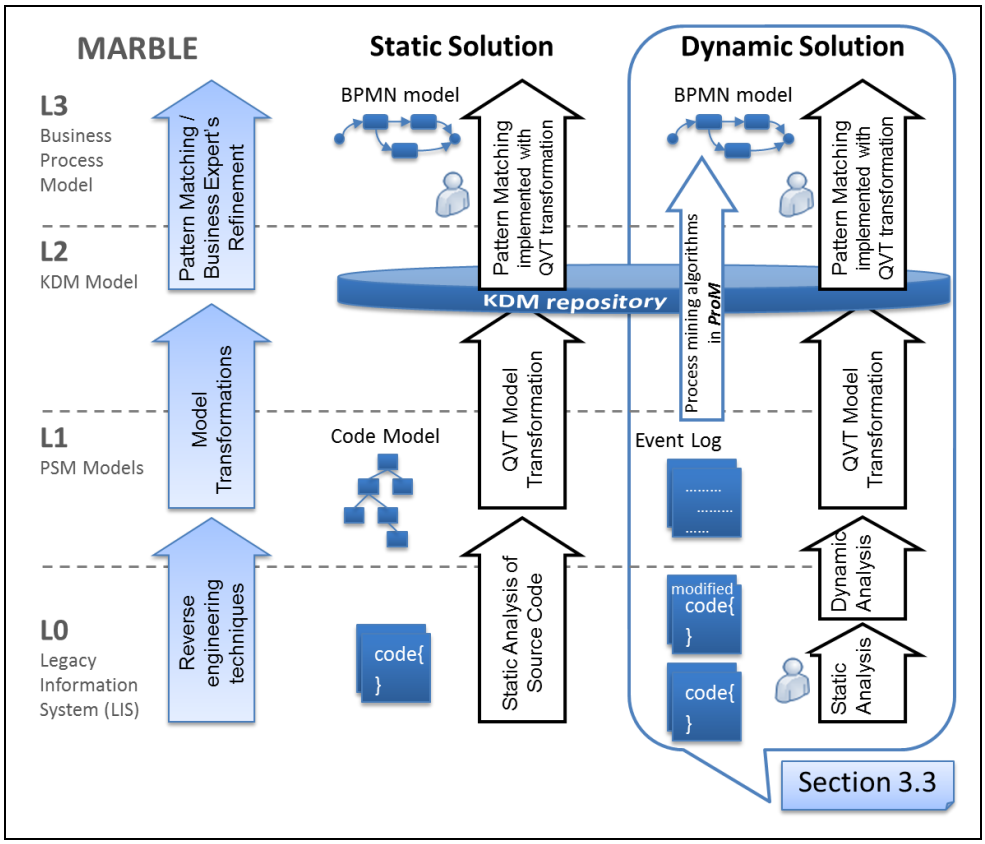


Figure 11. Main contribution of Section 3.3 as regards the proposal of the PhD Thesis as a whole

This section is supported by two journal papers. The first paper introduces a preliminary version of the dynamic technique. This paper provides a case study involving a real-life author management system. This paper was published by the IET Software Journal in 2011. This journal had an impact factor of 0.671 in 2010, according to Thomson's Science Citation Index (SCI). The reference of this paper is: *Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, and M. Piattini, Process Mining through Dynamic Analysis for Modernizing Legacy Systems. IET Software Journal, 2011. 5(3): p. 304–319.*

The second paper presents an improved version of the technique and validates the proposal by means of an industrial case study involving a healthcare evaluation system used in some

Austrian hospitals. This paper provides a case study involving a real-life author management system. This paper was published by the Enterprise Information System Journal in 2011. This journal had an impact factor of 0.786 in 2010, according to Thomson's Science Citation Index (SCI). The reference of this paper is: *Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, and M. Piattini, Generating Event Logs from Non-Process-Aware Systems Enabling Business Process Mining. Enterprise Information System Journal, 2011. 5(3): p. 301–335.*

www.ietdl.org

Published in IET Software
Received on 2nd September 2010
Revised on 15th December 2010
doi: 10.1049/iet-sen.2010.0103

ISSN 1751-8806

Process mining through dynamic analysis for modernising legacy systems

R. Pérez-Castillo[1] B. Weber[2] I.G.-R. de Guzmán[1] M. Piattini[1]

[1]*Alarcos Research Group, University of Castilla-La Mancha, Paseo de la Universidad, Ciudad Real 413071, Spain*
[2]*University of Innsbruck, Technikerstrasse 21a, Innsbruck 6020, Austria*
E-mail: ricardo.pdelcastillo@uclm.es

Abstract: Information systems age over time and become legacy information systems which often embed business knowledge that is not present in any other artefact. The embedded knowledge must be preserved to align the modernised versions of the legacy systems with the current business processes of an organisation. Modernisation efforts to preserve business knowledge typically consider different software artefacts as knowledge sources (e.g. code, databases, documentation etc.). Usually, the business knowledge needed to modernise a respective legacy system is statically recovered by reverse engineering techniques. Unfortunately, there is much knowledge that is only known during system execution. This study provides a semi-automatic technique based on dynamic analysis, combined with static analysis to instrument the source code for obtaining event log models. The event log represents a mapping between the pieces of source code executed and the business activities that they support. The obtained event log can then be used to mine the business processes embedded in legacy systems. In addition, the feasibility of the technique is validated by means of a formal case study, using a real-life legacy information system. The case study reports that the technique makes it possible to obtain event logs to effectively and efficiently discover business processes.

1 Introduction

Business processes have become a key asset in organisations, since these processes allow them to know and control their daily performance, and to improve their competitiveness [1]. A business process depicts a set of activities performed in an organisation that jointly realise a business goal [2]. These descriptions provide a way to map business objectives regarding how to best do operations within organisations. Therefore business processes also support the development of information systems to achieve these business objectives [3], that is, business processes are usually the starting point for developing information systems and form part of requirement analysis [4].

However, as a consequence of uncontrolled maintenance, enterprise legacy systems age over time. Thus, information systems significantly resist modification and evolution to meet new and constantly changing business requirements, becoming legacy information systems [5]. The progressive ageing and erosion of legacy information systems means that system maintainability diminishes below acceptable limits. Therefore legacy information systems must be modernised [6]. Modernisation means that the legacy system is replaced by a new one by means of re-implementation using another, better platform or an enhanced design, while the business knowledge of the system is preserved [7].

Business knowledge preservation requires an in-depth understanding of how the legacy information system currently supports the organisation's business processes. For this reason, the business processes implicitly described in a legacy information system must be recovered. Process mining has become a powerful tool for dealing with business process recovery [8] taking system execution into account. This means that process mining techniques and algorithms use the event log (i.e. a file containing all execution events) as the input artefact. This paper proposes the use of a process mining approach to recover and preserve business processes throughout modernisation projects. The main advantage of the process mining approach in contrast to other existing approaches is that it considers the dynamic perspective of legacy systems.

Unfortunately, process mining focuses on process-aware information systems [9], that is, process management systems such as enterprise resource planning (ERP) or customer relationship management systems. The nature of these systems (in particular their process-awareness) facilitates the registration of event logs throughout process execution. Indeed, most process mining techniques and tools are developed for this kind of information system [1]. In addition to process-aware information systems, however, a large number of traditional information systems also support the business processes of an organisation, and could thus benefit from process mining. Nevertheless, non-process-aware systems entail some challenges for obtaining meaningful event logs, since process definitions are implicitly described in legacy code and, thus, it is not clear which events should be recorded in the event log.

© The Institution of Engineering and Technology 2011

This paper proposes a technique based on dynamic analysis (combined with a static pre-analysis for code instrumentation) to obtain event logs from non-process-aware systems. The event logs represent the events related to the underlying business processes that occur during system execution, and thus this artefact provides another valuable source of knowledge to understand what is actually going on in a legacy information system from a dynamic perspective. The main advantage of our proposal is that it analyses the legacy source code from a dynamic point of view [10]. Since a lot of valuable information exists that is only known during system execution, there is a significant potential for exploiting runtime knowledge as well. The novelty of this technique, based on dynamic analysis, is that it allows obtaining event logs from traditional legacy information systems, which were not designed to register these logs.

In addition, this paper provides a model transformation to transform the event log into another model following the new knowledge discovery metamodel (KDM) standard [11] to depict legacy information system, concerning its runtime viewpoint, which can be used in any software modernisation project. This standardised representation can provide additional meaningful knowledge during a process mining activity throughout a software modernisation project.

The main benefit of the proposal is that the entire legacy information system is not discarded during software modernisation processes, but it contains significant business knowledge buried in the source code and other software artefacts as a consequence of maintenance modification over time. Therefore when a legacy system is modernised, business knowledge preservation can be addressed by means of the proposed technique, which achieves at least three advantages: (i) the embedded business knowledge that might not be present anywhere else is recovered [12]; (ii) the recovered business knowledge helps to align the modernised system and the current business processes of the organisation and (iii) the return of investment of the legacy information system is improved since the lifespan of the legacy system can be extended.

The feasibility of the proposed technique is validated by means of a case study involving a medium-size author management system. The study was rigorously conducted following a formal protocol for conducting case studies [13], thus in the future, the study might be replicated and the obtained results might also be compared. The empirical study focuses on the validation of effectiveness and efficiency of the proposed technique. After the execution of this case study, the study's results showed that the technique allows to obtain event logs that can be used to discover business processes with an adequate accuracy level. Furthermore, the study showed that the technique needs linear time with respect to the system size, thus the technique could be scaled to larger legacy information systems.

The remainder of this paper is organised as follows. Section 2 discusses related work. Section 3 presents the proposed technique based on dynamic analysis of source code to obtain event logs. Section 4 provides a model transformation to represent the event logs using KDM. Section 5 presents a case study involving a real-life legacy information system for author management. Section 6 provides conclusions and discusses future work.

2 Related work

Business process recovery is a common challenge that has been addressed in the literature for many years by both business experts and software engineers. Some works discuss business process recovery by means of registering event logs from process-aware information systems. For instance, van der Aalst *et al.* [14] provided different business process mining algorithms to discover business processes from event logs. This work compares some mining algorithm in industrial cases studies, which reports different values of effectiveness and efficiency for each algorithm. Moreover, the work presented by Günther *et al.* [15] provides a generic import framework for obtaining event logs from different kinds of process-aware information systems, although the efficiency of that proposal can be very different depending on the format of the information of each kind of process-aware information system. Ingvaldsen and Gulla [16] focused on ERP systems to obtain event logs from the SAP's transaction data logs. Despite that technique is processing-intensive, it achieves an intermediate value of effectiveness with a moderate efficiency.

Other approaches are aimed at traditional information systems without any built-in logging features, making different reverse engineering techniques necessary. For instance, several works address business process recovery in the context of non-process-aware information systems using static analysis. Zou and Hung [17] developed a framework to recover workflows from legacy information systems. This framework statically analyses the source code and applies a set of heuristic rules to discover business knowledge from the source code. Pérez-Castillo *et al* [18] made another proposal based on static analysis that uses a set of business patterns to discover business processes from source code. Ghose *et al.* [19] used text-based queries for extracting business knowledge from documentation. Paradauskas and Laurikaitis [12] recovered business knowledge by means of the inspection of the data stored in databases together with legacy application code. Wang *et al.* [20] presented a framework for business rules extraction from large legacy information systems based on static program slicing. Furthermore, do Nascimento *et al.* [21] presented a method for rewriting legacy systems based on business process management, but it provides a framework that consists of a manual business process recovery. All these approaches rely solely on static analysis or similar reverse engineering techniques, which has the disadvantage that a lot of knowledge is lost since some specific knowledge is only discovered at runtime. Those proposals do not need any expert's information at the beginning unlike our technique, thus all static-analysis-based proposals have a good efficiency level while the effectiveness is not as high as values obtained in other kinds of proposals.

Other solutions based on dynamic analysis have been suggested. Cai *et al.* [22] proposed an approach that combines requirement reacquisition with dynamic analysis. First, a set of use cases is recovered by interviewing the system's users. Secondly, the system is dynamically traced based on these use cases to recover business processes. Di Francescomarino *et al.* [23] provided a technique for recovering business processes by means of the dynamic analysis of the web application graphical user interface forms that are executed by the user throughout the navigation. This work additionally provides a clustering algorithm to minimise the obtained business processes. Another work, proposed by Ackermann *et al.* [24], presented a technique using dynamic analysis to recover business knowledge based on application-level views of the interaction between heterogeneous systems. In general, all techniques based on dynamic analysis, like our proposal, are more

© The Institution of Engineering and Technology 2011

time-consuming than static approaches, although the achieved effectiveness is usually higher.

In all these works, the technique for recovering event logs is restricted to a specific process mining algorithm. In contrast, our solution proposes a more generic technique based on dynamic analysis, which represents event logs according to the KDM standard at the end of applying the technique. Thus, different process mining algorithms can be applied to the resulting event logs to modernise legacy information systems as definitive goal.

3 Proposed technique to obtain event logs

The proposed technique provides the means to obtain event logs from legacy information systems, which contain a sequence of events representing the business activities executed when using the respective piece of source code. The obtained event logs can then be used to recover business processes. Event logs are commonly used in the process mining field as the input for several mining algorithms to discover the business process of an organisation. Usually, event logs are obtained from process-aware information systems [9]. However, the proposed technique deals with traditional (non-process-aware) information systems and thus it involves at least the following five key challenges:

Challenge C1. Missing process-awareness: Process definitions are implicitly described in legacy code. While process-aware information systems manage processes that consist of a sequence of activities or tasks with a common business goal using explicit process descriptions [2], traditional information systems consist of a control flow graph implicitly representing the business process that it supports (see Fig. 1). Thus, it is not obvious which events (related to a specific business activity) should be recorded in the event log, and in addition which parts of source code support a certain business activity. To address this challenge, the technique considers the 'a callable unit/a business activity' principle [17], which considers every callable unit (i.e. methods, functions, procedures etc.) as a candidate business activity in a process mining context.

Challenge C2. Granularity: While some of them are large callable units of a traditional information system that supports the main business functionalities of the system, many callable units are very small and do not directly support any business activity (e.g. setter and getter methods, printer methods etc.). To prevent the mined business processes from becoming bloated with unnecessary details, too fine-grained callable units should not be considered as activities in the event log, but must be discarded. Unfortunately, the set of callable units cannot easily be divided into coarse- and fine-grained callable units, since the threshold between these subsets is unknown, and it therefore entails another important challenge.

Challenge C3. Discarding technical code: Legacy source code not only supports business activities, but also technical aspects which have to be discarded when the event log is obtained. However, how we can know whether or not a callable unit belongs to the solution domain. As a first approximation, callable units in charge of auxiliary or technical functions that are not related to any use case of the system can be automatically discarded (e.g. callable units in charge of database access or the presentation of user interfaces). However, owing to the delocalisation and interleaving problems [25], to determine the technical and non-technical units is not an easy task (i.e. a domain package can contain some technical units or a technical package can contain some domain code). Therefore specific information provided by system analysts could help to address this challenge.

Challenge C4. Process scope: In contrast to the PAISs, traditional information systems do not explicitly define processes, and the information on where a process starts and ends cannot be automatically derived from the source code. Therefore both business experts and system analysts must provide this information to establish the scope of the business processes supported in the information system.

Challenge C5. Process instance scope: Each business process supported in the information system can be executed many times (i.e. a business process has several process instances in the event log). However, it is not clear how business activities and the multiple instances of a process should be correlated when the units are executed. It must be established which objects can be used to uniquely identify a process instance of a certain business process. However, the location of these objects in each callable unit has to be determined to correlate business activities obtained from those callable units. System analysts also help to solve this challenge by providing this information.

The technique proposed for obtaining an event log is based on a static pre-analysis of the source code combined with a dynamic analysis. First, the static analysis examines the legacy source code and modifies it by injecting additional code for writing after specific events during its execution. This stage is also known as code instrumentation (cf. Section 3.1). The static analysis is supported by specific information provided by business experts and system analysts. After the static analysis has been conducted, the instrumented source code is dynamically analysed

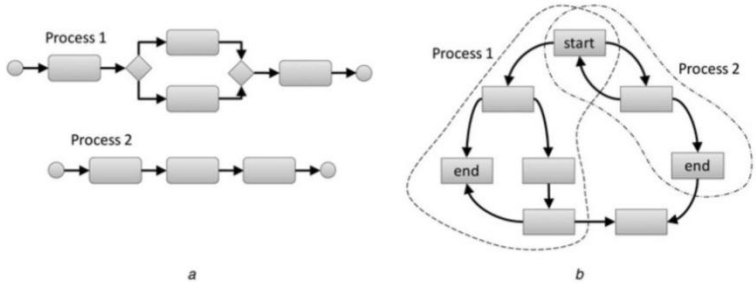


Fig. 1 *Comparison between PAIS and traditional information systems*
a Process-aware information system
b Traditional information system

© The Institution of Engineering and Technology 2011
IET Softw., 2011, Vol. 5, Iss. 3, pp. 304–319
doi: 10.1049/iet-sen.2010.0103

www.ietdl.org

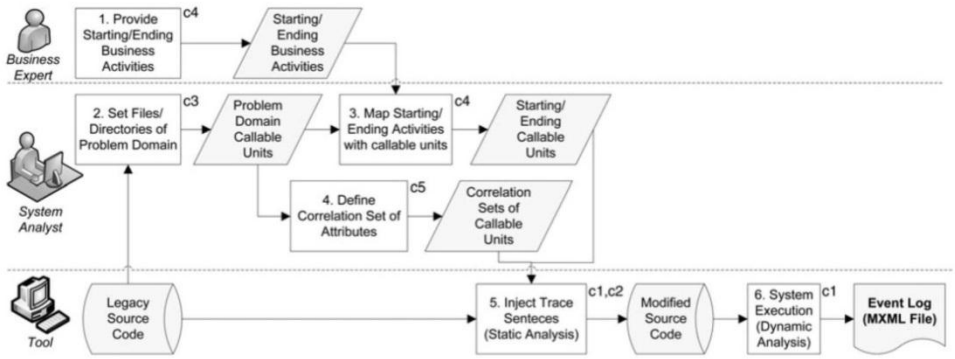


Fig. 2 *Overall process carried out by means of the proposed technique*

at runtime by means of the injected sentences (cf. Section 3.2). Fig. 2 presents an overview of the technique, the tasks carried out with their inputs/outputs and the challenge addressed for each task.

3.1 Code instrumentation through static analysis

The legacy source code is instrumented by means of static analysis, which modifies the original source code in a non-invasive way (i.e. small changes without affecting the behaviour and performance of the system) to enable the registration of events during system execution (see Fig. 2). To address the previously introduced challenges, code instrumentation by means of static analysis is supported with information provided by business experts and system analysts. In Task 1, business experts establish the start and end business activities of the business processes to be discovered (Challenge C4). In parallel, in Task 2, system analysts examine the legacy source code and filter the domain set of the directories, files or specific callable units that support business activities. This information is used to reduce potential noise in the runtime model due to the technical source code (Challenge C3). Task 3 consists of the mapping between start/end business activities and the callable units supporting them by system analysts (Challenge C4). In addition, through Task 4, system analysts establish the callable unit's arguments supporting the correlation data set which is uniquely identifying a process instance (Challenge C5).

The information which has to be provided by business experts and system analysts in the context of Tasks 1–4 is formalised according to the metamodel presented in Fig. 3. Both business experts and system analysts provide all the necessary information through a supporting tool, which generates a project with all that information. Each code instrumentation project is represented as a model that conforms with the code instrumentation metamodel presented in Fig. 3. The root metaclass is CodeInstrumentationProject which has two sub-metaclasses to respectively represent information provided by business experts and system analysts. Business expert information consists of the name of business processes (the BusinessProcess metaclass) and a set of business activities (the Activity metaclass), which is specialised into start and end activities. The SystemAnalystInfomation metaclass groups (i) the domain files or directories (a set of instances of the File metaclass); (ii) the match entries that link each business activity with a CallableUnit instance; (iii) the correlation data set represented by means of the CorrelationObject metaclass; and finally (iv) a set of strings to filter technical callable units (the Filter metaclass).

Finally, Task 5 carries out the syntactic analysis of the source code. On the fly, a parser analyses and injects the sentences for writing the event log during system execution. Fig. 4 shows the algorithm performed in Task 5 to automate the injection of sentences. During code instrumentation, the parser breaks down the source code

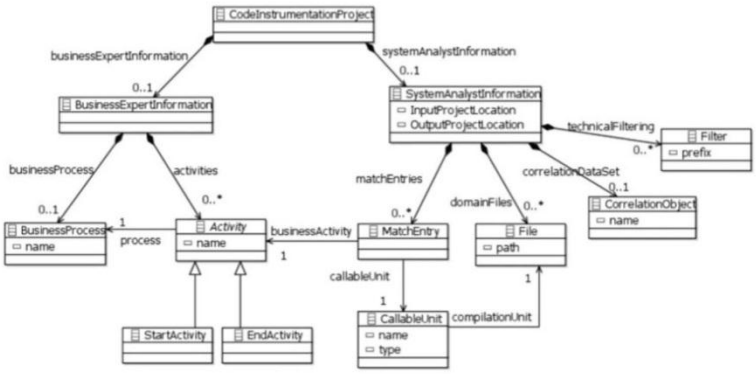


Fig. 3 *Metamodel to formalise and collect all the information of the code instrumentation stage*

© The Institution of Engineering and Technology 2011

www.ietdl.org

```
injectTraces (CallableUnits, DomainCallableUnits, StartingCallableUnits, EndingCallableUnits)
  ModifiedCallableUnits ← ϕ
  c' ← null
  For (c ∈ CallableUnits)
    If (c ∈ DomainCallableUnits and isFineGrainedUnit(c))
      If (c ∈ StartingCallableUnits)
        position ← 'first'
      Else If (c ∈ EndingCallableUnits)
        position ← 'last'
      Else
        position ← "intermediate"
      End-If
      sentence1←"writeEvent (c.name, 'start', position, c.correlationSet)"
      sentence2←"writeEvent (c.name,'complete',position, c.correlationSet)"
      c'.signature ← c.signature
      c'.body ← sentence1 + c.body + sentence2
      ModifiedCallableUnits ← ModifiedCallableUnits ∪ {c'}
    Else
      ModifiedCallableUnits ← ModifiedCallableUnits ∪ {c}
    End-If
  End-For
  Return ModifiedCallableUnits
```

Fig. 4 *Algorithm to code instrumentation by means of static analysis*

into callable units (Challenge C1). The algorithm then modifies the units of the domain set selected by system analysts in Task 3 (Challenge C3) and discards all other callable units. In addition, fine-grained callable units (e.g. setter, getter, constructor, toString and equals callable units) are automatically discarded (Challenge C2). After that, two sentences are injected at the beginning and the end of each filtered callable unit. The first sentence writes a start event related to the business activity mapped to the callable unit. It is injected between the signature and the body of the callable unit. The second sentence writes an end event for the respective business activity and is injected at the end of the body. Both sentences have additional parameters like the correlation data defined for the unit and the information whether or not the unit represents a start or end activity. This additional information is used when the injected sentences invoke the writeEvent function at runtime, which writes the respective event into the runtime model (cf. Section 3.2).

3.2 Dynamic analysis to obtain event logs

After having modified the source code through static analysis, it is released to production. The new code makes it possible to obtain an event log of the legacy information system represented according to the MXML format [15], which is used in the process mining field.

Fig. 5 shows the MXML metamodel, which provides the WorkflowLog metaclass to represent an event log as a set of instances of the Process metaclass. Each Process element contains several ProcessInstances, which have a sequence of AuditTrailEntriy elements. Each AuditTrailEntry element

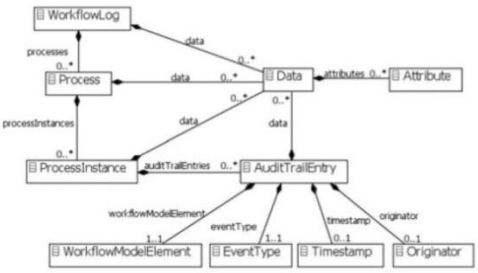


Fig. 5 *MXML metamodel used to represent runtime models in MARBLE*

represents an event and consists of four main elements: (i) the WorkflowModelElement that represents the executed activity; (ii) the EventType that represents whether the activity is being executed (start) or was completed (complete); (iii) the Originator which provides the user who starts or completes the activity; and finally (iv) the Timestamp which records the date and time of the event. Moreover, all these elements can have a Data element including additional information endorsed into Attribute elements.

Dynamic analysis is automatically carried out during system execution. Thus, when the control flow of the modified legacy information system reaches an injected sentence, a new event is added to the event log. The events are written by means of the writeEvent function.

Before adding the new event representing a business activity to the event log, it is necessary to find out the correct process and process instance where the event must be added. The appropriate process and process instance are located by means of Xpath expressions. If the process is null, then a new process is created. In addition, these expressions take the correlation data into account to determine the correct process instance. The attributes that contain the correlation data were already established during code instrumentation, although their values are only known during system execution.

Finally, when the writeEvent function has determined the correct process instance, it adds the event to that particular instance. The event, represented as an AuditTrailEntry element in the runtime model according to the MXML metamodel, is created using: (i) the name of the executed callable unit that represents the WorkflowModelElement; (ii) the event type that is also a parameter of this function; (iii) the user of the system that executed the callable unit (or the user of the session if the system is a web application), which represents the originator element; and finally (iv) the system date and time when the callable unit was executed to represent the timestamp element.

Fig. 6 shows an example of an MXML model obtained by applying the proposed technique. That model is represented according to the MXML metamodel. Fig. 6 illustrates, as an example, a partial view of the model provided by the tool developed to support the technique. The model is stored as an XML file, thus the developed tool presents the MXML model using a tree view. The partial view of the model represents an event log file, which contains two processes. Fig. 6 focuses on the first process named 'User Management' that contains at that moment a sole process instance named 'UserMgmtInstance_1'. That process instance shows four expanded events (AuditTrailEntry instances): 'searchAuthors' and 'editAuthor', both with the start and end TytpeEvent feature. In addition, the third expanded event has selected an Attribute instance within a Data element, which has the name 'correlationSet' and the value 'Author17354' (see Fig. 6). This information corresponds to the correlation data set for that particular process instance. For instance, a certain callable unit can be invoked several times; however, each execution of a particular callable unit is transformed into an event that is registered in a particular process instance according to the correlation data set.

During system execution, certain coverage must be ensured to obtain an event log that considers at least an appropriate number of execution paths. The proposal considers the function coverage (which is commonly used in software testing [26], among other research fields). This criterion

© The Institution of Engineering and Technology 2011
IET Softw., 2011, Vol. 5, Iss. 3, pp. 304–319
doi: 10.1049/iet-sen.2010.0103

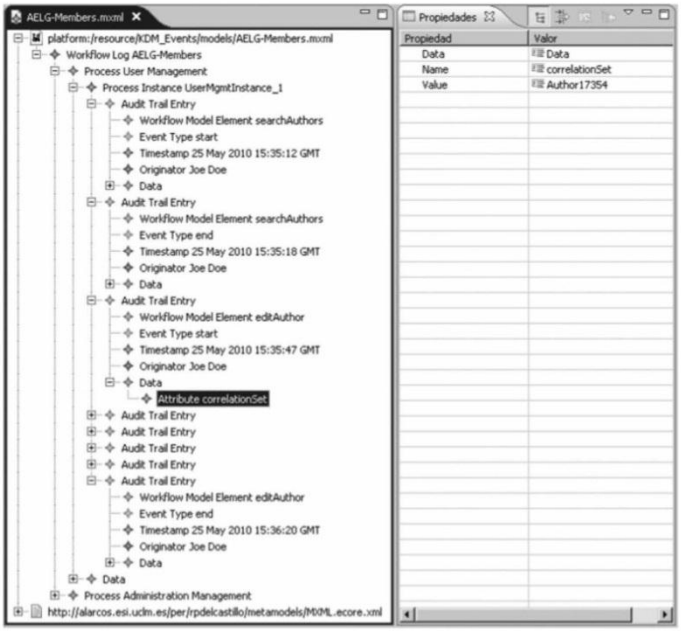


Fig. 6 *MXML model obtained by applying the technique with an author management system*

counts the callable units that are reached or invoked during system execution. The proposal considers this criterion instead of other criteria (e.g. sentence coverage, decision coverage or condition coverage), since the callable unit is the analysis unit considered according the 'a callable unit/a business activity' principle. Beside the coverage criterion, the limit to be used in each case must be established. For instance, a coverage value of 0.9 means that at least the 90% of all callable units were invoked during system execution. On the one hand, the coverage limit can be established at the beginning by business experts. On the other hand, the execution time can be previously fixed, and the coverage obtained after that time at the end of the dynamic analysis stage can be used as a significance value of the obtained results.

Ideally, the coverage limit must be a value close to 100%, however, it is not possible to reach a 100% of coverage in practice (at least in a reasonable time, since the coverage evolution is usually characterised with an asymptotic curve). Even executing the instrumented system during a long time, it does not ensure that a coverage value of 100% can be reached. This is due to the fact that, for instance, a bad legacy system design could not allow executing all callable units implemented in a legacy system (this problem is known as dead code). Thereby, a function coverage limit close to 100%, for many legacy information systems, will probably imply wait forever. According to the annual survey provided by Standish Group [27], about 50% of the features of a typical information system are not used or wasted. For this reason, our proposal considers a coverage limit of 65%.

As a result of the dynamic analysis stage, event logs are obtained which can be used as the input for different process mining algorithms. The business processes obtained using the obtained event log provides a good starting point to understand the source business process. If the obtained results do not provide sufficient quality, a new iteration of code instrumentation and dynamic analysis can be done. Both business experts and system analysts can refine the information provided for code instrumentation in the next iteration. As a consequence, the event log obtained during dynamic analysis could be better than the previous one. In addition, both kinds of experts might stop the iterations when the coverage percentage obtained in a concrete iteration has been reached. In this way, quick iterations (by using a lower coverage limit) can be done at the beginning to have progressively a better understanding of the business processes. After that the next iterations with an improved code instrumentation stage would generate more complete event logs. As a consequence, the iterative process becomes iterative and incremental, and the performance would be therefore better.

4 Event logs in KDM

Currently, the re-engineering concept that was typically used to re-implement legacy systems has shifted to so-called architecture-driven modernisation (ADM) [28] as a solution to the formalisation and standardisation problems of traditional reengineering. ADM advocates carrying out re-engineering processes following model-driven architecture (MDA) principles [29], that is, treating all the legacy software artefacts as models, and establishing model transformations between the different MDA abstraction levels. In addition, ADM defines the KDM standard, which has also been recognised as ISO standard 19506 [11]. KDM provides a common repository structure that makes it possible to exchange information about existing software artefacts involved in legacy information systems. KDM can be compared with the unified modelling language (UML)

© The Institution of Engineering and Technology 2011

standard, while UML can be used to generate new code in a top-down manner. ADM-based processes involving KDM start from the legacy source code and build a higher level model in a bottom-up manner [30].

This paper provides a model transformation between the model representing the event log at a lower abstraction level and a KDM model at higher abstraction level. This step is needed when the event log is used as an additional source of business knowledge in a software modernisation project. As a consequence, this model transformation increases the scope of the technique making it possible to use the technique in modernisation scenarios in addition to business process mining scenarios.

Specifically, the runtime models are represented using the event metamodel package of the runtime resource layer of the KDM metamodel [11]. Fig. 7 shows the event metamodel package as well as other metaclasses of other KDM packages used in the runtime models. The EventModel metaclass represents the runtime model, which contains a set of EventResource and EventAction elements. An EventResource element can be specialised into a state element, a transition element, an event element, or it can even be a container for other EventResources. The event element is used to model the AuditTrailEntries of the runtime model represented according to the MXML metamodel. The feature name represents the WorkflowModelElement, and the feature kind represents (with a 'start' or 'complete' value) the EventType.

The transformation is formalised by means of QVT-Relations, the declarative language of the query/view/transformation (QVT) standard [31]. A relation transforms an MXML model into an instance of the EventModel metaclass. This relation calls to the relation 'auditTrailEntry2Event' which transforms each AuditTrailEntry in the MXML model into an Event in the KDM model (see Fig. 8).

The event metamodel package of KDM makes it impossible to represent the process and process instance where the event belongs as well as the originator and timestamp of the event (see Fig. 7). For this reason, the proposal uses the default extension mechanism of the KDM metamodel: the extension families. The EventModel includes an ExtensionFamily element (see Fig. 8), which defines four Stereotype elements: <process>, <processInstance>,

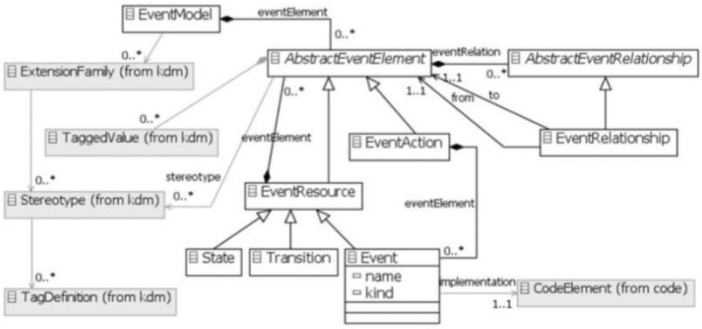


Fig. 7 *Event metamodel package in the KDM standard*

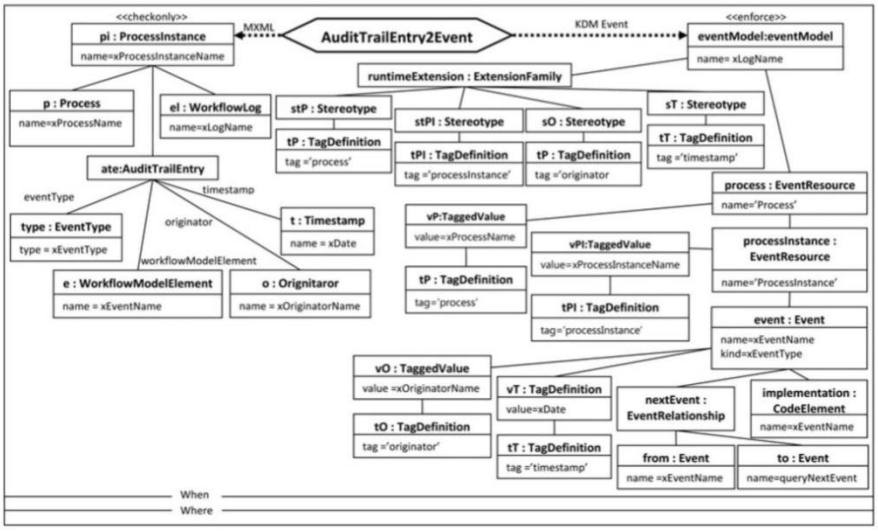


Fig. 8 *QVT relation 'auditTrailEntry2Event' to transform MXML models into KDM event model*

© The Institution of Engineering and Technology 2011
IET Softw., 2011, Vol. 5, Iss. 3, pp. 304–319
doi: 10.1049/iet-sen.2010.0103

<originator> and <timestamp >. Each stereotype has a TagDefinition element that is used by stereotyped elements of the KDM event model to put the specific value by means of a respective TaggedValue element. Therefore the problematic elements are represented in KDM as follows: (i) a process is represented as an EventResource element annotated with the <process> stereotype and containing a TaggedValue with the name of the process; (ii) a process instance is also represented as an EventResource element, which is nested within another EventResource that represents a process. This EventResource is annotated with the <processInstance> stereotype and contains a TaggedValue with the process instance identification; (iii) the originator is represented as a TaggedValue associated with an Event stereotyped as <originator >; and (iv) the timestamp is also represented with a TaggedValue in an Event annotated with the <timestamp> stereotype.

Despite the fact that the order of the events can be derived from the timestamp information, the 'auditTrailEntry2Event' relation (see Fig. 8) establishes the same sequence of events in the KDM event model as that registered in the MXML model. This order is represented as an EventRelationship within each Event representing a reference to the next Event element.

Finally, the 'auditTrailEntry2Event' relation (see Fig. 8) maps each Event to a CodeElement of the KDM code model. The resource runtime layer of the event package is above the program element layer (which contains the code and action metamodel packages), and thus the elements of a KDM event model can be mapped to the callable units represented in the KDM code model. As a consequence, the feature location is also improved throughout the modernisation of a legacy information system.

Fig. 9 provides a partial view of the KDM event model obtained from the MXML model that was obtained from an author management system (see Fig. 8). That KDM event model was obtained by applying the proposed QVT transformation. The event model has the same name as the MXML model, and contains first of all an extension family with the set of necessary stereotypes. After that, each process was transformed into an event resource using the <process> stereotype. The first process is named 'User Management', which contains another event resource using the <processInstance> stereotype, which has the same name than the process instance in the MXML model. That process instance has a set of Event elements, which contain the name and kind as a feature as well as a set of nested elements like (i) an annotation with the correlation data set; (ii) two tagged values that are stereotyped to represent the originator and timestamp information; and finally (iii) an EventRelationship element that links the current event with the next registered event (as can be seen after comparing Figs. 8 and 9).

5 Case study

This section presents a case study to validate the proposed technique by applying it to a real-life legacy information system. An *ad hoc* tool has been developed to semi-automatically support the proposed technique. The event logs are then obtained by means of dynamic analysis during the execution of the modified system. In addition, the ProM tool [32] (a process mining tool) is used to analyse the obtained event logs.

Moreover, the case study was carried out following the protocol for planning, conducting and reporting case studies proposed by Brereton *et al.* [13], improving the rigor and validity of the study. The following sections present the details of the main stages defined in the formal protocol: background, design, case selection, case study procedure, data collection, analysis and interpretation and validity evaluation.

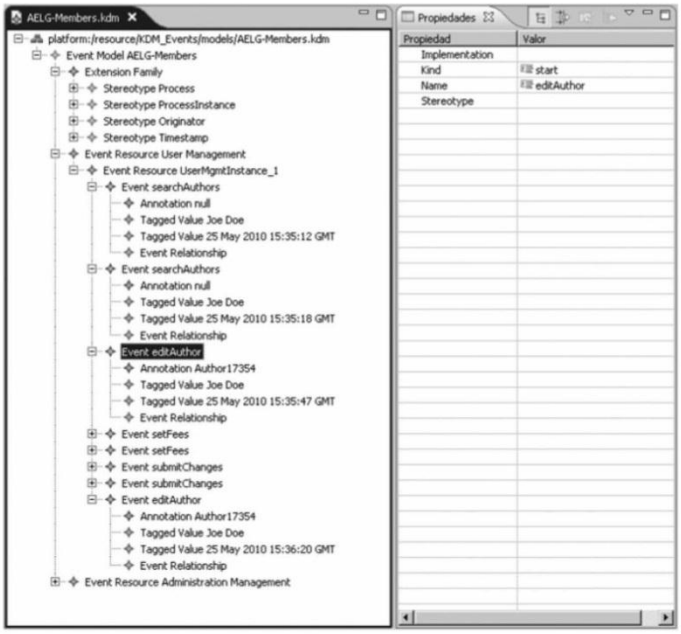


Fig. 9 *KDM event model obtained from the MXML model concerning an author management system*

© The Institution of Engineering and Technology 2011

5.1 Background

First, the previous research on the topic must be identified. The related work presented in Section 2 discusses other proposals for recovering business processes from legacy information systems and compares them with our proposal. Particularly, our proposal focuses on non-process-aware systems and is based on a dynamic analysis of the source code to obtain events logs. Therefore the object of this study is that the proposed technique obtains event logs from non-process-aware information systems, and the purpose of this study is the evaluation of specific properties of the proposed technique related to its effectiveness and efficiency.

Taking into account the object and purpose of the study, it defines two main research questions (MQ) (see Table 1). First, MQ1 checks if the technique can obtain event logs dealing with the presented challenges, and the obtained event logs allow process mining tools to effectively discover business processes. MQ1 evaluates whether the business processes recovered from the event log accurately represent the business behaviour of the organisation that owns the legacy information systems. In addition, Table 1 shows two additional research questions (AQ) that are derived from MQ1, and make it possible to answer MQ1. The first sub-question AQ1 is established to evaluate whether the discovered business processes comprise all the elements that are present in the organisation's current business processes. The second sub-question AQ2 checks whether the processes mined from the obtained event log contain any elements that do not belong to the business processes of the organisation. Secondly, MQ2 is related to the efficiency of the proposed technique to evaluate if it can be used with any information system despite of its size. MQ2 is also divided into two additional questions. AQ3 evaluates the time spent on the injection of the special sentences in the source code of the legacy information system, and AQ4 evaluates what is the performance penalty induced by the source code modifications.

5.2 Design

The study follows the holistic case study design according to the classification proposed by Yin [33]. This means that the case study consists of a single case, that is, it focuses on a single legacy information system. In addition, the case is studied as a whole, in that the study does not consider several analysis units within the case. The study consists of the application of the proposed technique to obtain an event log, which is then mined to discover a set of business processes that is analysed to answer the research questions established in Table 1. For this purpose, some measures are established to quantitatively answer the questions (see Table 2).

To evaluate the effectiveness of the proposed technique through question MQ1, the study proposes to use two measures: precision and recall. These measures are usually used in information retrieval scenarios [34], and we adapted these measures for a business process recovery scenario. These measures are used because precision can be seen as a measure of exactness or fidelity, whereas recall is a measure of completeness. On the one hand, the measure precision (M1 in Table 2) represents the amount of relevant recovered elements within the set of recovered elements in a business process model. An element is considered relevant if this element faithfully represents business operations or the business behaviour of the organisation in the real world. The measure precision is used to answer question AQ1. On the other hand, the measure recall (M2 in Table 2) represents the amount of relevant recovered elements of the total of relevant elements (recovered and not recovered) that depict the whole business operation of the organisation. The measure recall is related to the answer to question AQ2. While each discovered business process is the independent variable, these two measures are dependent variables in the study.

The study considers the task element as the score unit to apply these measures in a business process recovery

Table 1 Case study research questions

Id	Research question
MQ1	can the proposed technique obtain event logs from legacy information systems to effectively mine business processes?
AQ1	can the technique obtain event logs discovering all the elements of the embedded business processes?
AQ2	can the technique obtain event logs without discovering any elements not belonging to the business processes?
MQ2	is the proposed technique efficient to be scaled to any legacy information system?
AQ3	is the time spent on the source code modification linear with regard to the size of the legacy information system?
AQ4	is the information system's performance affected by the source code modifications?

Table 2 Case study measures

Id	Measure	Formula	Research question
M1	precision	$P = \frac{\lvert\{\text{relevant tasks}\} \cap \{\text{recovered tasks}\}\rvert}{\lvert\{\text{recovered tasks}\}\rvert}$	AQ1
M2	recall	$R = \frac{\lvert\{\text{relevant tasks}\} \cap \{\text{recovered tasks}\}\rvert}{\lvert\{\text{relevant tasks}\}\rvert}$	AQ2
M3	*F*-measure	$F = \frac{2\text{PRECISION} \times \text{RECALL}}{\text{PRECISION} + \text{RECALL}}$	MQ1
M4	performance time	$T = T_{\{\text{Manual Intervention}\}} + T_{\{\text{Code Instrumentation}\}} + T_{\{\text{Dynamic Analysis}\}}$	AQ3
M5	ratio of injected sentences	$R_S = \frac{\text{Injected Sentences}}{\text{Lines of Source Code}}$	AQ4

© The Institution of Engineering and Technology 2011
IET Softw., 2011, Vol. 5, Iss. 3, pp. 304–319
doi: 10.1049/iet-sen.2010.0103

scenario. Therefore precision (M1) is defined as the number of true relevant tasks divided by the total number of relevant (recovered) tasks, that is, the sum of true relevant tasks and false relevant tasks that were incorrectly recovered. Moreover, recall (M2) is defined as the number of true relevant tasks divided by the total number of relevant tasks, that is, recovered and not recovered relevant tasks of the business process.

We also use the business expert opinion to discover which recovered tasks are or are not relevant to evaluate the proposed measures. In this respect, the definition of relevant task is a key factor in facilitating the business experts' work, and it must be defined a priori. Despite the fact that the evaluation of measures concerns only the business tasks, the relevant task definition implicitly considers other business elements. The relevant task definition defines a set of four conditions. Condition C1 specifies that the task must represent a real-life business operation within the organisation. This condition is not evaluated by considering task names, since these names are inherited from legacy code and they may be biased as regards the real business activity names provided by business experts. Condition C2 ensures that all the tasks preceding the evaluated task must be recovered relevant tasks. To obtain this condition, the predecessor tasks cannot be directly related to the evaluated task through intercalated non-relevant tasks. Condition C3 ensures that all the subsequent tasks must be directly (or indirectly) recovered relevant tasks. Finally, Condition C4 specifies that all the object data related to the evaluated task have been recovered.

Although the precision and recall measures are adequate, there is an inverse relationship between them. As a consequence, extracting conclusions to answer MQ1 with an isolated evaluation of these measures is very difficult. For this reason, these measures are usually combined into a single measure known as F-measure (M3), which consists of a weighted harmonic mean of both measures.

Moreover, another two measures are considered to evaluate the MQ2 and its sub-questions related to the efficiency of the technique under study. M4 (see Table 2) specifies the total time spent by the tool on the injection of the sentences in the legacy source code including the manual intervention, as well as the time spent on dynamic analysis. This measure is related to the answer of the AQ3. In addition, the time penalty when the modified source code is executed should be measured. However, this task is not easy, since there are several functionalities and services in an information system which can be executed in different scenarios. Thus, the response time for each information system' service could be biased if the service or functionality is not executed under the same conditions. As a consequence, the study uses the fraction of sentences injected in the source code (M5 in Table 2), to have an approximation of the performance penalty due to the dynamic analysis during system execution.

5.3 Case selection

Case selection is a key stage in case study planning, which aims to select a good and suitable case to be studied. Table 3 presents the five criteria established to select the most appropriate information system. Criterion Cr1 guarantees that the selected legacy system is a real-life information system that supports the business operation of an organisation or company (e.g. it discards embedded systems or real-time systems). Criterion Cr2 ensures that the legacy system would be a traditional system and not have any built-in logging mechanism. Cr3 ensures that the selected system really is a legacy information system. The time in production is not a good measure to check this criterion. Indeed, Cr3 considers the amount of modifications in the system that alter the source business processes (i.e. adaptive and perfective modifications [35]). Criterion Cr4 ensures that the system is large enough to draw representative conclusions, thus it fixes the limit in 10 000 lines of source code. Finally, criterion Cr5 guarantees that the system is based on the Java platform, since the tool supporting the technique at this stage is only available for Java-based systems. However, the proposed technique is not specific for Java-based systems, but is based on the concept of callable units which can also be applied to other programming languages (i.e. method in case of Java, procedure in case of C or COBOL etc.). For this reason, the tool is currently being upgraded by implementing parsers for additional programming languages.

Table 3 Criteria for case selection

Id	criterion for case selection
Cr1	it must be an enterprise system
Cr2	it must be a non-process-aware information system
Cr3	it must be a legacy system
Cr4	it must be of a size not less than 10 KLOC
Cr5	it must be a Java-based system

After evaluating several available systems according to the above criteria, the legacy information system selected for study was AELG-members, which supports the administration of an organisation of Spanish authors. The system automates several services offered by the organisation, including, among others, author registration, cancellation of memberships and payment of fees. For this reason, the system meets Cr1. In addition, AELG-members does not have any mechanism to register event logs, thus it also meets Cr2. Moreover, the first release of AELG-members was moved to the production stage six months ago. During this time the system had one medium modification (version 1.1), a large modification (version 2.0) and two more medium modifications (versions 2.1 and 2.2). Thus, the system's Cr3 compliance can also be ensured. From a technological point of view, AELG-members is a Java application meeting Cr5, and its architecture follows the traditional structure into three layers [36]: (i) the domain layer supporting all the business entities and controllers; (ii) the presentation layer dealing with the user interfaces; and (iii) the persistency layer handling data access. The total size of the legacy system is 23.5 KLOC ensuring Cr4.

5.4 Case study procedure

After design and selection of the case study, the study's execution procedure must also be planned. The execution is partially aided by the tool developed to support the proposed technique. The case study procedure defines the followings steps:

1. After some meetings between the staff of the candidate organisations and researchers, the legacy information system is selected according to the case selection criteria. In addition, the business expert and system analyst from the organisation who will participate in the study are appointed in this step. In addition, the selected business expert is

© The Institution of Engineering and Technology 2011

needed to perform the manual post verification to evaluate the measures M1 and M2.

2. The legacy source code is modified by means of the developed tool which is used by the business expert and system analysts according to the proposed technique (cf. Section 3). The intervention of business experts and system analysts in the proposed technique may seem a complex and time-consuming task. However, the design of the tool makes it possible to easily and quickly collect the information needed.

3. The instrumented source code of the legacy system under study is implanted in the organisational environment, that is, the source code is implanted in a host, the database schema is built through the database scripts, the initial data is loaded and so on. After that, the instrumented system is used by some of the usual users to progressively generate an event log. The instrumented system is executed until the function coverage reaches a value above 65%.

4. The event log obtained by applying the technique is analysed by means of the ProM tool to discover a set of preliminary business processes. ProM is the world-leading tool in the area of process mining [37], which support a lot of techniques and algorithms to discover processes from event logs. We use in this study the genetic mining algorithm, since it is the algorithm (focusing in the extracting sequence flow structure) that has demonstrated the best accuracy [37]. The genetic algorithm was executed with a population size of 100 and a maximum number of generations of 1000.

5. The preliminary business processes obtained from the analysis of the event log are modified by the business experts. They fit the preliminary business processes with the reality of the organisation. Business expert intervention is also used to evaluate the proposed measures. Business experts annotate which task elements in the obtained business processes are accurately recovered and which are wrong. In addition, business experts identify the tasks that should have been recovered but were not recovered.

6. In parallel to steps 4 and 5, the model transformation is executed to obtain a KDM event model from the MXML event log. The QVT transformation is executed using *Medini QVT* [38], a QVT engine to execute QVT relations.

7. All key information related to the generation of the event log (steps 2 and 3), the discovered business processes (step 4), the business expert intervention (step 5), as well as the model transformation (step 6), is collected according to the data collection plan (see Section 5.5).

8. The data collected in the previous step is analysed and interpreted to draw conclusions to answer the research questions. Section 5.6 presents the results obtained from this case study.

9. Finally, the case study is reported and feedback is given to the organisation and the research community.

5.5 Data collection

The data to be collected and the data sources must be defined before starting the execution of the case study to ensure the future repeatability. First, the business expert's configuration to obtain the event log is recorded (see Table 4), which was collected by means of the tool. Table 4 also shows the manual intervention time.

Table 5 shows the data collected through the code instrumentation after static analysis, which presents: (i) the total number of source code lines; (ii) number of source code files in AELG-members; (iii) number of *Java* files; (iv) the number of sentences injected; and (v) total time to obtain the modified source code. All this information is provided by the tool after the modification of source code.

Table 4 Business expert information to aid the code instrumentation

Process	Start activity	End activity
categories management	search categories	save categories
author management	search author edit author	save author
reporting	search author search authors	print report
correlation data set: the parameter ' '*idAuthor*' in different callable units		
manual intervention time = 88′34″		

Table 5 Code instrumentation results

Feature	Value
LOC	23522
# source code files	364
# *Java* source code files	165
# injected sentences	1946
static analysis time, s	178.34″

After dynamic analysis when the event log has been obtained, a summary of the data in the event log is also collected using *ProM*, a tool used to discover the business processes. Table 6 shows (i) the total processes registered in the event log; (ii) the number of process instances; (iii) the total number of events; (iv) the mean of events per process instance, (v) the function coverage established; as well as, (vi) the time that the modified system was executed to obtain a meaningful event log taking into account the coverage degree; and finally Table 6 shows the time spent on executing the MXML-to-KDM transformation.

Finally, Table 7 shows the data obtained from the business process discovery with the ProM tool and the business expert intervention to evaluate the precision and recall measures. Table 7 shows (i) the number of recovered tasks (before manual intervention); (ii) the number of recovered relevant tasks (i.e. the number of tasks that the business experts mark as correct); (iii) the number of recovered non-relevant tasks (i.e. tasks removed from the business process since they do not represent a business activity); (iv) the precision and (v) the recall values for each final business process; and (vi) the harmonic mean between both measures.

5.6 Analysis and interpretation

After the data have been collected, it is analysed to draw the conclusions. The analysis should obtain the evidence chains

Table 6 Event log analysis

Feature	Value
# processes	3
# process instances	3780
# events	51360
mean of events per instance	14
function coverage	65%
generation time	47 days
transformation time, ms	100152

© The Institution of Engineering and Technology 2011
IET Softw., 2011, Vol. 5, Iss. 3, pp. 304–319
doi: 10.1049/iet-sen.2010.0103

www.ietdl.org

Table 7 Final business process model data

Business process model name	# Recovered tasks (RcT)	# Recovered relevant tasks (RcRvT)	# Recovered non-relevant tasks (RcNRvT)	# Non-recovered relevant tasks (NRcRvT)	Precision (RcRvT/ RcRvT + RcNRvT)	Recall (RcRvT/ RcRvT + NRcRvT)	F-Measure (2•P•R)/(P + R)
categories management	11	7	4	2	0.636	0.778	0.700
author management	23	12	11	2	0.522	0.857	0.649
reporting	6	4	2	1	0.667	0.800	0.727
mean	13.3	7.7	5.7	1.7	0.608	0.812	0.692
standard deviation	8.7	4.0	4.7	0.6	0.08	0.04	0.04

from the data to answer the research questions. The obtained event log has a total of 3780 events with an average of 14 events per process instance (see Table 6). The event log comprises three processes in total, which are related to the three processes established at the beginning of the code instrumentation stage (see Table 4). Fig. 10 shows as an example the final 'Author Management' business process after the business expert post-intervention. This preliminary business process was obtained using the genetic algorithm of ProM, which was discovered in 2.3 min. The preliminary 'Author Management' business process had 23 recovered tasks, of which 12 were relevant tasks and 11 non-relevant tasks. Moreover, two additional relevant tasks ('Import Authors' and 'Approve Changes') were added by business experts after the business process discovery from the event log (see Fig. 10).

The outgoing event log was obtained by executing the modified information system for 47 days (i.e. until the function coverage reached a level of 65%). Fig. 11 provides the function coverage evolution during execution of the AELG-members system. The first execution day achieved 25% of function coverage. At the beginning, when the event log is empty and no callable unit has been executed, function coverage quickly grows. At the end, when a lot of callable units have already been executed, the coverage percentage increases more gradually (see Fig. 11).

To answer the research question MQ1, the values of the precision and recall measures were evaluated (on average for the three business processes) with values of 0.61 and 0.81, respectively. On the one hand, a higher recall value means that the proposed method recovers a higher number

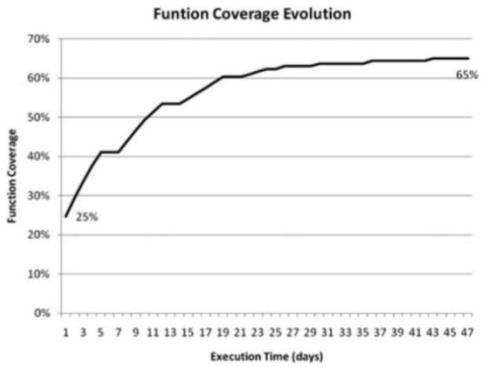


Fig. 11 *Function coverage evolution during system execution*

of relevant business elements. On the other hand, a lower precision value contrasts with the high recall value, which means that a large amount of non-relevant tasks has been recovered. In most cases, the significant amount of non-relevant tasks is due to the respective tasks did not represent business activities, but rather technical source code (e.g. auxiliary methods), or may be the tasks were classified as non-relevant due to their inappropriate level of granularity (e.g. their business knowledge is already included in other larger relevant tasks).

In any case, the results obtained are usual since there is an inverse relationship between precision and recall measures.

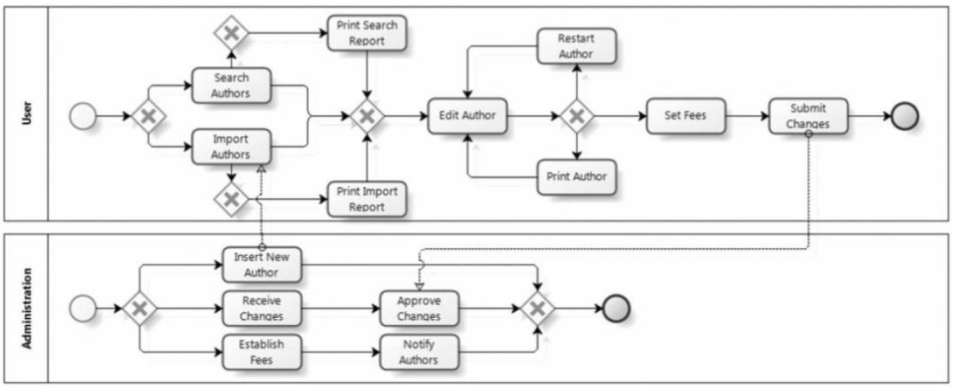


Fig. 10 *Final 'author management' business process*

© The Institution of Engineering and Technology 2011

Indeed, the precision value should, ideally, always be 1.0 for any recall value but, according to [39], this is not possible in practice. Therefore owing to the fact that the relationship between both measures has an inverse nature, it is possible to increase one at the cost of reducing the other. This means that the proposed method could reduce its recall value by recovering fewer tasks, at the cost of reducing the number of non-relevant recovered tasks, that is, increasing the precision value. This hypothetical result is more desirable than the obtained result, since the precision and recall values would be more balanced. In fact, the F-measure can be considered as a special case of the general F_α-measure (1) where α is 1. The α value is used to weight the precision in relation to recall, and the selected F_1-measure (M3 in Table 2) thus signifies that both precision and recall are equally important. In this study, the F_1-measure had an average value of 0.692.

$$F_\alpha = \frac{(1+\alpha) \times \text{PRECISION} \times \text{RECALL}}{\alpha \times \text{PRECISION} + \text{RECALL}} \quad (1)$$

To answer AQ1 and AQ2, and in consequence MQ1, the obtained values were additionally compared with reference values from other experiences with model recovery in the literature, such as those of [40–42]. We found reports of precision and recall values close to 60%, and these were our benchmark values. The values obtained for our measures (precision = 61%, recall = 81% and F_1-measure = 69%) were therefore slightly above 60%, the reference value. Thereby, sub-questions AQ1 and AQ2 can be answered positively, thus the proposed MQ1 (see Table 1) is evaluated as true, that is, the proposed technique can obtain event logs from legacy information systems to effectively mine business processes.

Moreover, to answer MQ2 the sub-questions AQ3 and AQ4 must be answered. The time spent on modifying the original source code $T_{\{\text{Code Instrumentation}\}}$ was 178 s (see Table 5). In addition, the technique involves two additional time penalties. First, $T_{\{\text{Dynamic Analysis}\}}$ as a tiny performance penalty due to the injected sentences execution. This penalty is constant with respect to the system size, since it only affects the response time of each system's service or functionality in particular. For this reason $T_{\{\text{Dynamic Analysis}\}}$ can be considered as a negligible value. Secondly, the time spent on the manual intervention of business experts and system analysts to provide the needed information, which was $T_{\{\text{Manual Intervention}\}} = 89$ min. As a result, the final time value was $T = 5340 + 178 + 0$ s. The technique's bottleneck is obviously the manual intervention for a sole static analysis. However, this fact is not a disadvantage of the technique for two main reasons. First, several iterations are possible further improving the quality of the event log. In the subsequent iterations, the manual intervention effort is progressively reduced until it is not needed, thereby $T_{\{\text{Manual Intervention}\}}$ decreases when the number of code instrumentation iterations grows. Secondly, the proposed technique is better than business process redesign by business experts from scratch, since those solutions are more time-consuming and error-prone than semi-automatic techniques.

As a consequence, to evaluate AQ3, the analysis focuses on the time of the automatic component. The total time was 92 min which seems feasible for the selected case, since the size of the system is 23.5 KLOC. Nevertheless, the scalability of the technique must be evaluated according to question AQ3. For this purpose, we captured the partial time spent on the code instrumentation by static analysis of each java package. A linear regression model was considered with the time as the dependent variable and the size (lines of source code) of each java package as the independent variable. Fig. 12 shows the scatter chart of size/time showing the regression line, which presents a positive linear relationship between the package size and the time spent on modifying this package. The time variable in the regression model considers the partial time $T_{\{\text{Code Instrumentation}\}}$ for each java package. To compare time values through the regression model, it does not consider $T_{\{\text{Manual Intervention}\}}$ for the dynamic solution, since this time it only represents an increase in the vertical axis while the slope of the regression line is not affected. In addition, $T_{\{\text{Dynamic Analysis}\}}$ can be considered as a negligible value.

Moreover, Fig. 12 shows the correlation coefficient of the proposed regression model (R^2), which is the degree to which the real values of the dependent variable are close to the predicted values, that is, how much points are fitted to the regression line. In this study, the correlation coefficient was 0.81, and owing to the fact that the R^2 value is between 0 and 1 for a positive linear relationship, this value is relatively high since it is close to 1. The proposed linear regression model is therefore suitable to explain the results of the case study; that is, there is no quadratic or exponential relationship between time and size. The expected increase in time for larger systems will consequently be linear, and the time will thus be assumable. Question AQ3 can therefore be answered as true.

Moreover, to answer the last sub-question AQ4 the ratio of injected sentences (R_S) was considered in this study. The total injected sentences value was 1946, and the total number of sentences in the system is 23 522, thus the R_S ratio was 8.2%, a low percentage. In addition, since an injected sentence is quite simple (i.e. it only executes a search and writing of a fragment in an XML file) its execution is not time-consuming. Therefore the system's performance (due to the dynamic analysis) is not affected by the source code modifications significantly, and the question AQ4 is evaluated as true. In any case, the tiny performance penalty owing to this fact is constant with respect to the system size, since it only affects the response time of each system's service or functionality. As a consequence, the second main

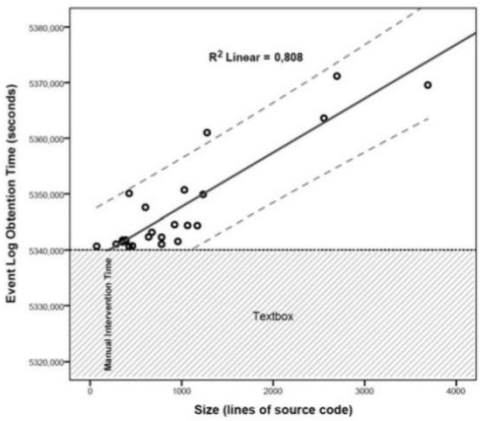


Fig. 12 *Size/time scatter chart*

© The Institution of Engineering and Technology 2011
IET Softw., 2011, Vol. 5, Iss. 3, pp. 304–319
doi: 10.1049/iet-sen.2010.0103

research question MQ2 is also evaluated positively. This means that the proposed technique can be efficiently scaled to any legacy information system.

Finally, the MQ3 related to the model transformation to obtain KDM event models must be answered. First, AQ5 is evaluated to know if the MXML-to-KDM transformation effectively obtains KDM models. AQ5 considers the transformation completeness measure (M6 in Table 2). The transformation completeness was 97.14% (see Table 6), which is a high percentage. This fact means that only 2.86% of the expected number of metaclass instances was not transformed in the outgoing KDM event model. The effectiveness of the model transformation is therefore ensured, that is, AQ5 can be positively answered. Moreover, the transformation time is also evaluated to answer AQ6. The total transformation time was 1.6 min for an event log with 51 360 events. This time is lower than, for example, the time spent on obtaining business processes from event log using the genetic algorithm (29.1% less); or it is also lower than the time spent on code instrumentation stage (53.9% less). These comparisons only mean that the time is reasonable to use the model transformation; however, the actual efficiency of the transformation cannot be evaluated without any reference value or comparison with other study results. In any case, the AQ6 can be answered as true, and as a result, MQ3 is also positively answered.

5.7 Validity evaluation

Finally, the validity of the results must be evaluated, that is, if the results are true and not biased for the whole population for which we want to generalise the results. This section shows the threats to the validity of this case study as well as the list of possible actions to mitigate them. There are mainly three types of validity: internal, construct and external.

5.7.1 Internal validity: First, there are two threats to the internal validity. The first threat is related to the tools used to instrument the legacy source code (the developed tool), to discover the business processes (ProM), as well as to execute the model transformation (Medini QVT). The specific tools affect the obtained results; the measured values might be different if other tools are used. To mitigate this threat, the study could be replicated using different tools and compared with the obtained results.

The second threat is that the results concerning the precision and recall measures could also be biased due to the manual intervention by business experts, since it represents a subjective point of view. This threat is difficult to eradicate, however, different business experts teams could be considered to have different viewpoints.

5.7.2 Construct validity: According to the construct validity, the proposed measures were adequate to measure the variables and answer the research questions appropriately. The precision and recall measures were reused from the information retrieval field, where these measures have an adequate maturity level. In addition, these measures allow us to check whether the business processes obtained accurately represent (or not) the business behaviour of the organisation. Nevertheless, the reference value around 50% taken from the literature to compare the obtained results may be quite relative. Unfortunately, there are not enough benchmark values for these metrics in the process mining field, thus this thread is not easy to mitigate.

5.7.3 External validity: Finally, external validity is concerned with the generalisation of the results. This study considers traditional legacy information systems as the whole population. In this respect, the obtained results could be generalised to this population. However, the specific platform of the selected case is a threat that should be noted. Thus, the results are strictly extended to those legacy and non-process-aware information systems based on Java language. However, the expected results for other kinds of object-oriented systems could be similar to the results of this case study. To mitigate this threat, the study should be replicated using information systems based on a different platform, and the results obtained in these studies should be compared to have a better understanding about the performance of the proposed technique.

6 Conclusions and future work

Software modernisation projects typically take into account several software artefacts as sources of knowledge, but these artefacts are analysed in a static way. However, a dynamic approach allows modernisation projects to extract more meaningful knowledge. Thus, this paper proposes a technique to obtain event models by means of the dynamic analysis of source code. First, the proposed technique statically analyses the legacy source code and modifies it by injecting special sentences that make it possible to register execution events. This stage is supported by business experts and system analysts. Secondly, during system execution, the modified code is dynamically analysed through the execution of the source code modified with the injected sentences and an event log model is written according to the MXML metamodel. The event log depicts the sequence of executed events related to the business activities of the business processes embedded in the legacy source code.

In addition, the event log model can be transformed into a KDM event model according to the KDM standard. For this purpose, an extension of the event package of the KDM metamodel is proposed as well as a model transformation implemented by means of QVT-Relation. Owing to the fact that the event log is represented following the KDM standard, the proposed technique can be used in any modernisation framework based on ADM.

One of the most important topics of discussion is that the performance of the technique can seem infeasible since it needs a lot of information provided by business experts and system analysts. However, the manual intervention of the proposed technique is not really a complex and time-consuming task. On the one hand, the manual intervention is supported by means of a tool that alleviates this task. On the other hand, the manual intervention necessary in our technique (the start point to discover business processes) is much less than the manual intervention necessary when the business experts are modelling the business processes of an organisation from scratch. In addition, if the business experts draw the process from scratch, they probably do not take the knowledge embedded in the legacy information system into account, and the business processes might be modelled with some deviations with respect to the organisation's information systems. Therefore we consider the manual intervention by both business experts and system analysts assumable in the proposed technique.

The analysis of source code is carried out in static way, that is, it analyses the source code file, but it does not analyse the execution of the source code. Static analysis is less intelligent

© The Institution of Engineering and Technology 2011

than dynamic analysis, since there is specific knowledge that is known only at run time. However, the dynamic analysis usually requires source code modifications to aggregate traces like the proposed technique, but it is not always possible since the legacy systems can be in the production stage. Thus, another point of controversy concerns the implementation of the proposed technique in a realistic environment, since it might be thought that a company or organisation would not accept the use of an automatically modified version of its information system. Nevertheless, the technique ensures that the injection of the special sentences (for writing event logs) is done without affecting the behaviour of the original information system, that is, in a non-invasive way.

In any case, the feasibility of the proposed technique has been empirically validated by means of a case study involving an author management system. The case study was conducted following a formal protocol, and it reports that the technique makes it possible to obtain event logs to effectively discover the current business processes of an organisation. In addition, the case study demonstrates that the technique can be efficiently used with larger information systems.

The planned work focuses on replicating the case study with a great variety of information systems to obtain strengthened conclusions. Indeed, the tool is currently being upgraded by implementing parsers for additional programming languages. The objective is to validate the technique by using different kinds of legacy information systems, that is, different program languages, diverse sizes of the systems, number of different business experts and system analysts, the availability of those experts, and so forth.

Moreover, future work will focus on using KDM event models combined with other models of KDM as the code and data models to obtain more meaningful business process models, since these models make it possible to consider additional sources of embedded knowledge. In addition, aspect-oriented programming could be considered to improve the proposed code instrumentation stage, since this approach could help to detect the different execution scenarios related to the aspects of a particular information system. Each execution scenario could be then combined with the 'a callable unit/a business activity' principle to recover events logs achieving more effectiveness.

7 Acknowledgments

This work was supported by the FPU Spanish Program; by the R + D projects funded by JCCM: ALTAMIRA (PII2I09-0106-2463), INGENIO (PAC08-0154-9262) and PRALIN (PAC08-0121-1374); and the PEGASO/MAGO project (TIN2009-13718-C02-01) funded by MICINN and FEDER. In addition, this work was supported by the University of Innsbruck.

8 References

1 Castellanos, M., Medeiros, K.A.d., Mendling, J., Weber, B., Weitjers, A.J.M.M.: 'Business process intelligence,' in Cardoso, J.J., van der Aalst, W.M.P. (Eds.): 'Handbook of research on business process modeling' (Idea Group Inc., 2009), pp. 456–480

2 Weske, M.: 'Business process management: concepts, languages, architectures (Leipzig, Alemania)' (Springer, Berlin, Heidelberg, 2007), p. 368

3 Jeston, J., Nelis, J., Davenport, T.: 'Business process management: practical guidelines to successful implementations' (Butterworth-Heinemann (Elsevier Ltd.), NV, USA, 2008, 2nd edn.), p. 469

4 Heuvel, W.-J.v.d.: 'Aligning modern business processes and legacy systems: a component-based perspective (Cooperative information systems)' (The MIT Press, 2006)

5 Brodie, M.L., Stonebraker, M.: 'Migrating legacy systems: gateways, interfaces, and the incremental approach' (Morgan Kaufmann, 1995)

6 Newcomb, P.: 'Architecture-driven modernization (ADM)'. Proc. 12th Working Conf. on Reverse Engineering, 2005

7 Khusidman, V., Ulrich, W.: 'Architecture-driven modernization: transforming the enterprise', DRAFT V.5, 2007, http://www.omg.org/docs/admtf/07-12-01.pdf, OMG, p. 7

8 van der Aalst, W., Weijters, A.J.M.M.: 'Process mining,' in Dumas, M., van der Aalst, W., Ter Hofstede, A. (Eds.): 'Process-aware information systems: bridging people and software through process technology' (Wiley, 2005), pp. 235–255

9 Dumas, M., van der Aalst, W., Ter Hofstede, A.: 'Process-aware information systems: bridging people and software through process technology' (Wiley, 2005)

10 Cornelissen, B., Zaidman, A., Deursen, A.v., Moonen, L., Koschke, R.: 'A systematic survey of program comprehension through dynamic analysis', *IEEE Trans. Softw. Eng.*, 2009, **35**, (5), pp. 684–702

11 ISO/IEC, ISO/IEC DIS 19506. Knowledge discovery meta-model (KDM), v1.1 (architecture-driven modernization), 2009, http://www.iso.org/iso/catalogue_detail.htm?csnumber=32625, ISO/IEC, p. 302

12 Paradauskas, B., Laurikaitis, A.: 'Business knowledge extraction from legacy information systems', *J. Inf. Technol. Control*, 2006, **35**, (3), pp. 214–221

13 Brereton, P., Kitchenham, B., Budgen, D., Li, Z.: 'Using a protocol template for case study planning'. Evaluation and Assessment in Software Engineering (EASE'08), Bari, Italia, 2008

14 van der Aalst, W., Reijers, H., Weijters, A.: 'Business process mining: an industrial application', *Inf. Syst.*, 2007, **32**, (5), pp. 713–732

15 Günther, C.W., van der Aalst, W.M.P.: 'A generic import framework for process event logs'. Business Process Intelligence Workshop (BPI'06), 2007, (*LNCS*, **4103**), pp. 81–92

16 Ingvaldsen, J.E., Gulla, J.A.: 'Preprocessing support for large scale process mining of SAP transactions'. Business Process Intelligence Workshop (BPI'07), 2008, (*LNCS*, **4928**), pp. 30–41

17 Zou, Y., Hung, M.: 'An approach for extracting workflows from e-commerce applications'. Proc. 14th Int. Conf. Program Comprehension, 2006, pp. 127–136

18 Pérez-Castillo, R., García-Rodríguez de Guzmán, I., Ávila-García, O., Piattini, M.: 'MARBLE: a modernization approach for recovering business processes from legacy systems'. Int. Workshop on Reverse Engineering Models from Software Artifacts (REM'09), Lille, France, 2009, pp. 17–20

19 Ghose, A., Koliadis, G., Chueng, A.: 'Process discovery from model and text artefacts'. IEEE Congress on Services (Services'07), 2007

20 Wang, X., Sun, J., Yang, X., He, Z., Maddineni, S.: 'Business rules extraction from large legacy systems'. Proc. Eighth Euromicro Working Conf. Software Maintenance and Reengineering (CSMR'04), 2004

21 do Nascimento, G.S., Iochpe, C., Thom, L.H., Reichert, M.: 'A method for rewriting legacy systems using business process management technology'. 11th Int. Conf. Enterprise Information Systems (ICEIS'09), Milan, Italy, 2009, pp. 57–62

22 Cai, Z., Yang, X., Wang, W.: 'Business process recovery for system maintenance – an empirical approach'. 25th Int. Conf. on Software Maintenance (ICSM'09), Edmonton, Canada, 2009, pp. 399–402

23 Di Francescomarino, C., Marchetto, A., Tonella, P.: 'Reverse engineering of business processes exposed as web applications'. 13th European Conf. on Software Maintenance and Reengineering (CSMR'09), Kaiserslautern, Germany, 2009, pp. 139–148

24 Ackermann, C., Lindvall, M., Cleaveland, R.: 'Recovering views of inter-system interaction behaviors'. Proc. 16th Working Conf. on Reverse Engineering, 2009, pp. 53–61

25 Ratiu, D.: 'Reverse engineering domain models from source code'. Int. Workshop on Reverse Engineering Models from Software Artifacts (REM'09), Lille, France, 2009, pp. 13–16

26 Ammann, P., Offutt, J.: 'Introduction to software testing' (Cambridge University Press, 2008), p. 322

27 The Standish Group: 'CHAOS summary 2010' (The Standish Group International, Inc., 2010)

28 OMG: ADM Task Force by OMG, 2007, 9/06/2009 [cited 2008, 15/06/2009]. Available at: URL

29 Miller, J., Mukerji, J.: 'MDA Guide Version 1.0.1, 2003, www.omg.org/docs/omg/03-06-01.pdf, OMG. 62

30 Moyer, B.: 'Software archeology. Modernizing old systems', *Embed. Technol. J.*, 2009, http://adm.omg.org/docs/Software_Archeology_4-Mar-2009.pdf

© The Institution of Engineering and Technology 2011

doi: 10.1049/iet-sen.2010.0103

31 OMG, QVT: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, 2008, http://www.omg.org/spec/QVT/1.0/PDF, OMG

32 van der Aalst, W.M.P., van Dongenm, B.F., Günther, C., Rozinat, A., Verbeek, H.M.W., Weijters, A.J.M.M.: 'ProM: the process mining toolkit'. Seventh Int. Conf. Business Process Management (BPM'09) – Demonstration Track, Ulm, Germany, 2009, pp. 1–4

33 Yin, R.K.: 'Case study research. Design and methods' (Sage, London, 2003, 3rd edn.)

34 Raghavan, V., Bollmann, P., Jung, G.S.: 'A critical investigation of recall and precision as measures of retrieval system performance', *ACM Trans. Inf. Syst.*, 1989, **7**, (3), pp. 205–229

35 ISO/IEC, ISO/IEC 14764:2006. Software engineering – software life cycle processes – maintenance, 2006, http://www.iso.org/iso/catalogue_detail.htm?csnumber=39064

36 Gamma, E., Helm, R., Johnson, R., Vlissides, J.: 'Design patterns: elements of reusable object-oriented software' (Longman Publishing Co. Inc., Addison-Wesley, Boston, MA, USA, 1995)

37 Medeiros, A.K., Weijters, A.J., Aalst, W.M.: 'Genetic process mining: an experimental evaluation', *Data Min. Knowl. Discov.*, 2007, **14**, (2), pp. 245–304

38 ikv++, Medini QVT. http://www.ikv.de/index.php?option=com_content&task=view&id=75&Itemid=77. 2008, (ikv + + technologies ag)

39 Davis, J., Goadrich, M.: 'The relationship between precision-recall and ROC curves'. Proc. 23rd Int. Conf. Machine Learning, Pittsburgh, Pennsylvania, 2006, pp. 233–240

40 Lucrédio, D., Fortes, R.P.M., Whittle, J.: 'MOOGLE: a model search engine'. 11th Int. Conf. Model Driven Engineering Languages and Systems, Toulouse, France, 2008, pp. 296–310

41 Ye, Y., Fischer, G.: 'Supporting reuse by delivering task-relevant and personalized information'. 24th Int. Conf. Software Engineering, Orlando, Florida, 2002, pp. 513–523

42 Garcia, V.C., Lucrédio, D., Durão, F.A., *et al.*: 'From specification to experimentation: a software component search engine architecture'. Ninth Int. Symp. on Component-Based Software Engineering (CBSE 2006), Västerås, Sweden, 2006, pp. 82–97

© The Institution of Engineering and Technology 2011

Enterprise Information Systems
Vol. 5, No. 3, August 2011, 301–335

Generating event logs from non-process-aware systems enabling business process mining

Ricardo Pérez-Castillo[a]*, Barbara Weber[b], Jakob Pinggera[b], Stefan Zugal[b], Ignacio García-Rodríguez de Guzmán[a] and Mario Piattini[a]

[a]*Alarcos Research Group, University of Castilla-La Mancha, Paseo de la Universidad 4, 13071 Ciudad Real, Spain;* [b]*Quality Engineering, University of Innsbruck, Technikerstraße 21a, 6020 Innsbruck, Austria*

(*Received 10 February 2011; final version received 8 May 2011*)

As information systems age they become legacy information systems (LISs), embedding business knowledge not present in other artefacts. LISs must be modernised when their maintainability falls below acceptable limits but the embedded business knowledge is valuable information that must be preserved to align the modernised versions of LISs with organisations' real-world business processes. Business process mining permits the discovery and preservation of all meaningful embedded business knowledge by using event logs, which represent the business activities executed by an information system. Event logs can be easily obtained through the execution of process-aware information systems (PAISs). However, several non-process-aware information systems also implicitly support organisations' business processes. This article presents a technique for obtaining event logs from traditional information systems (without any in-built logging functionality) by statically analysing and modifying LISs. The technique allows the modified systems to dynamically record event logs. The approach is validated with a case study involving a healthcare information system used in Austrian hospitals, which shows the technique obtains event logs that effectively and efficiently enable the discovery of embedded business processes. This implies the techniques provided within the process mining field, which are based on event logs, may also be applied to traditional information systems.

Keywords: process mining; event log; dynamic analysis; modernisation; legacy system

1. Introduction

Business processes have become a key asset in organisations, since they allow them to follow their daily performance and to address technological and organisational changes and consequently improve their competitiveness (Castellanos *et al.* 2009). Enterprise information systems automate most of the business processes of an organisation (Weske 2007) but uncontrolled maintenance causes them to age and become legacy information systems (LISs) (Ulrich 2002), e.g. owing to code written long ago becoming technologically obsolete. LISs significantly resist modification and evolution, failing to meet new and constantly changing business requirements

*Corresponding author. Email: ricardo.pdelcastillo@uclm.es

ISSN 1751-7575 print/ISSN 1751-7583 online

(Paradauskas *et al.* 2006). Their continuous evolution implies that their maintainability eventually falls below acceptable limits, meaning that they have to be modernised (Mens 2008), which may be done, for instance, by re-implementing them using another, better platform or an enhanced design, while the business knowledge embedded in them is preserved (Ulrich and Newcomb 2010). It is very important to take up the challenge of preservation of embedded knowledge for two main reasons: (i) it is not present in any other artefact, and (ii) it must be considered when aligning the improved system with the organisation's current business processes (Heuvel 2006).

Since a company's information system is usually the main repository of its business knowledge (Mens 2008), its modernisation requires an in-depth understanding of how it currently supports the organisation's business processes. This problem motivates the use of process mining, which has become a powerful tool for understanding what is really going on in an organisation by observing the information systems (van der Aalst and Weijters 2005). It seeks to extract business process knowledge from process execution logs known as *event logs*, which contain information about the start and completion of activities and the resources executed by the processes (Castellanos *et al.* 2009), and are the input for the majority of the process mining techniques and algorithms. There are several techniques in the field of process mining for recovering business processes from three different perspectives (van der Aalst and Weijters 2005): (i) the process perspective, focusing on the control flow between business activities; (ii) the organisational perspective, describing the organisational structure; and (iii) the case perspective, focusing on the characterisation of each execution of the process, also known as process instances.

Currently, process mining techniques and algorithms focus on event logs obtained from process-aware information systems (PAISs) (Dumas *et al.* 2005), e.g. process management systems such as enterprise resource planning (ERP) or customer relationship management (CRM) systems, whose nature (in particular their process-awareness) facilitates the direct registration of events throughout process execution. Indeed, most process-mining techniques and tools are developed for information systems of this kind (Castellanos *et al.* 2009), and the event logs are represented in a common format used by the majority of process-mining tools known as MXML (Mining XML) (van der Aalst *et al.* 2009). In addition to PAISs, a vast number of traditional (non-process-aware) information systems (most of which have become LISs) also support the business processes of an organisation, and could thus benefit from process mining.

However, such non-process-aware information systems impose five key challenges for obtaining meaningful event logs for feasible use with process-mining techniques: (i) business process definitions are implicitly described in legacy code, so it is not obvious which events should be recorded in the event log; (ii) there is often different granularity between callable units (invocable pieces of source code like methods, function, procedures, etc.) of an information system and activities of a business process; (iii) legacy code not only contains business activities, but also technical aspects that have to be eliminated when mining a business process; (iv) since traditional systems do not explicitly define business processes, their starts and ends have to be established; (v) finally, absence of process-awareness means that it is not obvious how business activities should be correlated in each case.

This article proposes a technique for obtaining process event logs from traditional (non-process-aware) information systems addressing these challenges

on the basis of both static and dynamic analyses of the systems' source code to simulate the event log registration mechanism of a PAIS. Firstly, the static analysis syntactically analyses the source code and injects pieces of it in a non-invasive way in specific parts of the LIS. Secondly, the dynamic analysis of the modified source code makes it possible to create an event log file in MXML format during system execution. The proposed technique is further supported by specific information that must be provided by business experts and systems analysts familiar with the system. Its main advantage is the analysis of the legacy source code from a dynamic point of view (Cornelissen *et al.* 2009), which means that it only considers valuable information gathered during system execution. Its novelty is the possibility of obtaining event logs from traditional LISs that were not designed to create them. Its feasibility is demonstrated with a case study involving a healthcare information system, whose results show that the event logs obtained are of good quality.

The remainder of this article is organised as follows. Section 2 discusses related studies. Section 3 presents an example used in subsequent sections to illustrate the challenges and the solution proposed. Section 4 explains the main challenges for obtaining event logs from traditional information systems. Section 5 presents the technique proposed to tackle them. Section 6 provides a case study of a healthcare system to validate the proposal. Finally, Section 7 provides a conclusion and discusses future work.

2. Related work

There are several studies in the literature that address the problem of preserving the embedded business knowledge in software modernisation processes in traditional (non-process-aware) information systems. Some are based on static analysis of source code like Zou and Hung (2006), who developed a framework to recover workflows from LISs. This framework statically analyses the source code and applies a set of heuristic rules to discover business knowledge from source code. Similarly, Pérez-Castillo *et al.* (2009) propose MARBLE, an architecture-driven modernisation (ADM) framework that uses the knowledge discovery metamodel (KDM) standard to obtain business processes by statically analysing legacy source code. In addition to source code, other works consider additional legacy software artefacts. For example, Paradauskas and Laurikaitis (2006) present a framework to recover business knowledge through the inspection of the data stored in databases. Ghose *et al.* (2007), in turn, propose a set of text-based queries in source code and documentation for extracting business knowledge. Motahari Nezhad *et al.* (2008) propose *Process Spaceship*, a static analysis-based technique to discover process views from process related data sources (e.g. web service logs). This approach addresses the event correlation challenge using correlation conditions defined as binary predicates over the event contents by business experts. Motahari *et al.* (2007) propose a similar approach that uses web service logs to deal with the event correlation challenge. However, this approach uses graph theory techniques instead of information provided by business experts.

All these studies rely solely on static analysis as a reverse engineering technique to extract business knowledge. However, they have the major disadvantage that run-time information is not taken into account. Therefore, other solutions have been suggested based on dynamic analysis, which has been applied for a wide set of topics in the literature (Cornelissen *et al.* 2009), for instance a feature location technique

(Eisenbarth *et al.* 2003), which gathers the information from a set of scenarios invoking the features. Other authors (Di Francescomarino *et al.* 2009) consider graphical user interfaces of web applications to discover business processes. In addition, Cai *et al.* (2009) propose an approach combining requirement reacquisition with dynamic and static analysis techniques. Firstly, a set of use cases is recovered by means of interviewing the system's users. Secondly, the system is dynamically traced according to these use cases. Finally, the traces obtained at run-time are statically analysed to recover business processes.

All these works based on dynamic analysis share a major limitation, since they always propose specific solutions focusing on a sole mining algorithm to discover specific business knowledge and use specific or proprietary formats to represent the processes or the intermediate information. As a consequence, these approaches cannot be reused in several mining scenarios since they consist of *ad hoc* solutions. In this regard, other approaches focus on obtaining event logs as an intermediate step. Event logs depict the sequence of business process activities executed and are then used in several well-known process mining algorithms to discover the embedded business knowledge. For instance, Günther and van der Aalst (2007) provide a generic import framework for obtaining event logs from different kinds of PAIS. It deals with some interesting challenges, some of them similar to those in this article, but they are related to PAISs rather than LISs. In addition, Ingvaldsen and Gulla (2008) focus on ERP systems to obtain event logs from the SAP's transaction data logs.

All these proposals have been suggested for use with PAISs, although traditional LISs have not been treated so far using the event log approach. Thus in contrast to these studies, our solution proposes a technique based on dynamic analysis (combined with static analysis) to obtain MXML event logs from traditional LISs that is not restricted to a specific process-mining algorithm. Table 1 shows a comparison between the different business process recovery techniques in the literature and our proposal. To compare all the techniques each proposal is framed

Table 1. Comparison of other business process recovery proposals with ours.

Perspective	Without event logs		Obtaining event logs	
Source of knowledg Web service logs	Static analysis	Dynamic analysis	Static analysis	Dynamic analysis
Business export information	Motahari Nezhad *et al.* (2008)	Serrour *et al.* (2008)		
Systems analyst information	Cai *et al.* (2009)		Our proposal	
Source code Documentation	Eisenbarth *et al.* (2003)	Zou *et al.* (2006), Pérez-Castillo *et al.* (2009)	Ghose *et al.* (2007)	Günther *et al.* (2007)
Database	Paradauskas *et al.* (2006)			
User interfaces		Di Francescomarino *et al.* (2009)		
SAP transaction data			Ingvaldsen *et al.* (2008)	

in the matrix of Table 1. The rows of the matrix represent all the different sources of knowledge (or software artefacts), and the columns specify the system perspective (static or dynamic) considered for extracting business knowledge.

In addition to all the effort regarding business process mining techniques and algorithms, there is work about the usage of process mining techniques in other areas apart from the business process management field. These approaches assume the presence of event logs and they are not concerned with obtaining event logs from systems without in-built logging functionality. For example, Funk *et al.* (2010) use process mining techniques to analyse data about product usage and user experience. Rozinat *et al.* (2009) investigate the applicability of process mining to less structured processes. Trkman *et al.* (2010) employ business process mining techniques to improve the supply chain in companies. Hofstra (2009) uses process mining techniques to get insights into the actual user behaviour and improve consequently the usability of software products. Günther *et al.* (2010) use process mining techniques to analyse the test process of ASML, which is the world's leading manufacturer of chip-making equipment and a key supplier to the semi-conductor industry.

3. A demonstrative example

This section introduces an example to illustrate all the challenges in Section 4 and to explain the working of the technique presented in Section 5. The example considers a small business process as the object of study, and a *Java* application implementing it. Figure 1 shows the source business process, which is based on the product order process described by *Weske* (Weske 2007). This process allows registered customers to place orders. In parallel, customers receive the products and the invoice.

Figure 2 shows the structure of the small application developed to support the business process illustrated in Figure 1. The application follows the traditional decomposition into three layers (Eckerson 1995): (i) the domain layer supports all the business entities and controllers; (ii) the persistence layer handles data access; and (iii) the presentation layer deals with the user interfaces (see Figure 2, left). The BuyerController class contains most of the logic of the application (see Figure 2, right), i.e. it provides the methods supporting the activities of the source business process. Figure 2 includes additional symbols over the screenshot to aid the explanation of the process-aware challenges and technique proposed in following sections.

4. Process-awareness challenges

This section shows the main challenges for obtaining event logs from traditional information systems: lack of process-awareness, granularity, elimination of technical code, process scope and process instance scope.

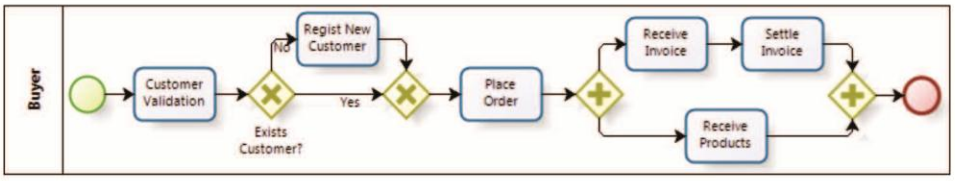


Figure 1. The original business process for ordering products.

4.1. Challenge 1 – lack of process-awareness

Ascertaining which activities are executed is the first major challenge to registering the events of a traditional (non-process-aware) information system, owing to the different natures of traditional information systems and PAISs. While PAISs manage processes consisting of a sequence of activities or tasks with a common business goal using explicit process descriptions (Weske 2007) (see Figure 3A), traditional systems are a set of methods, functions or procedures (*callable units* in general) where processes are only implicitly described and thus blurred. Traditional systems can be seen as a graph where nodes represent the callable units, and arcs represent their invocation (see Figure 3B). The call graph thus represents the control flow of a traditional system according to the domain logic implemented.

To address this challenge, Zou and Hung (2006) propose the '*a callable unit/a business activity*' approach which considers each callable unit of a traditional information system as a candidate business activity in a process mining context. Zou and Hung's approach provides a good starting point, but is of a heuristic nature, and therefore ignores several major challenges concerning the inherent nature of source code such as the different granularity of callable units and business activities (cf.

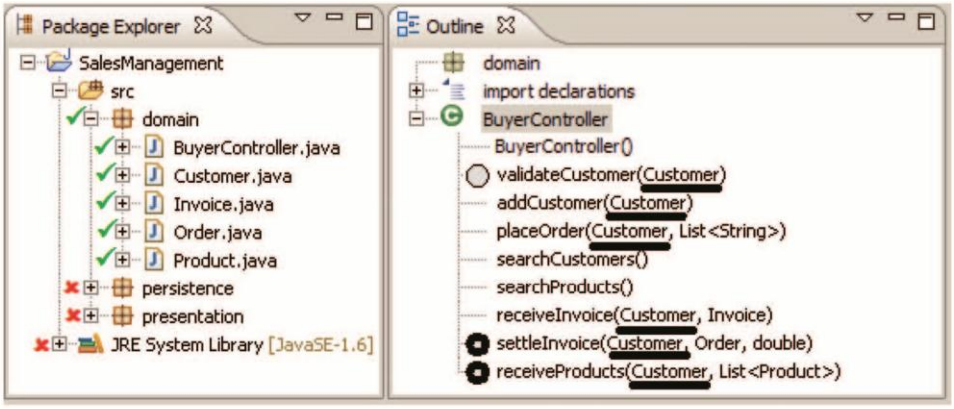


Figure 2. Structure of the legacy Java application.

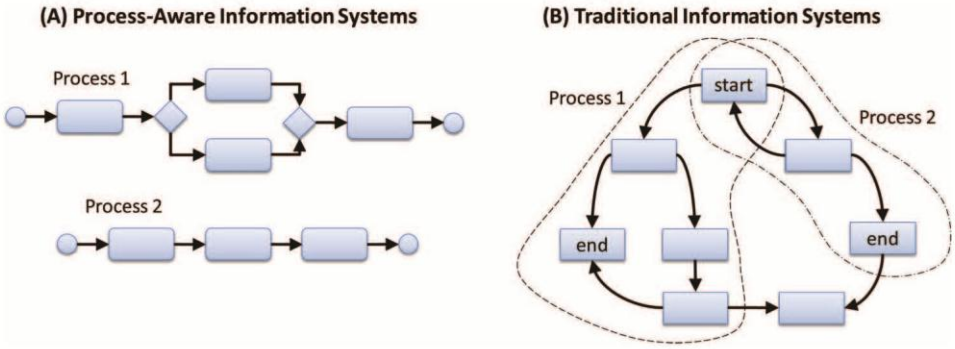


Figure 3. Comparison between PAISs and traditional LIS.

Section 4.2) and the mixture of business-related callable units and technical callable units typical of LISs (cf. Section 4.3).

4.2. Challenge 2 – granularity

The different granularity of business activities and callable units in LISs constitutes another important challenge. According to the approach proposed by Zou and Hung (2006), each callable unit in a traditional LIS is considered as an activity to be registered in an event log. However, traditional systems typically contain thousands of callable units, some of which are large ones supporting the main business functionalities of the system (case i), while many are very small and do not directly support any business activity (case ii). In some situations, a set of small callable units together supports a business activity (case iii).

Cases (i) and (ii) could be addressed by considering only coarse-grained callable units as business activities and discarding fine-grained ones, but, unfortunately, the dividing line between coarse- and fine-grained callable units is unknown. However, different solutions may be implemented to rule out fine-grained callable units, such as source code metrics (e.g. lines of source code metric or the cyclomatic complexity metric) (Rozman *et al.* 1991). This solution is easy to implement, but has the disadvantage of high computational costs when the event log file is written during run-time over a long period. Alternatively, heuristic-based solutions can offer a good alternative with minimal computational costs. For instance, all *setter* and *getter* methods of the classes representing business entities like *Customer* or *Product* (see Figure 2) only read or write object fields and can thus be considered fine-grained units (case ii). To avoid mined business processes getting bloated with unnecessary details, the fine-grained callable units are not considered as activities in the event log.

For case (iii), another heuristic can be defined to cluster some fine-grained callable units into a business activity. For instance, if all the *setter* methods of the *Customer* entity (see Figure 2) are executed consecutively for the same *Customer* object, all those executed callable units can be considered as a unique business activity named, for instance, '*create Customer*', dealing with creating new customers. As a consequence, those fine-grained callable units will not be discarded (despite the previous heuristic for cases (i) and (ii)), but a new business activity is created from all those fine-grained units executed in succession.

4.3. Challenge 3 – discarding technical code

Another important challenge is caused by the fact that LISs typically contain several callable units that cannot be considered business activities, but are of a technical nature. Callable units can be divided into two groups: (i) the *problem domain* group, which contains the callable units related to the business entities and functionalities of the system to solve the specific problem (i.e. those implementing the business processes of the organisation) and (ii) the *solution domain* group, containing the callable units related to the technical nature of the platform or programming language used and aiding the callable units of the previous group. Since callable units belonging to the solution domain do not constitute business activities, they should not be considered in the event log.

However, *how can we know whether or not a callable unit belongs to the solution domain?* As a first approximation, callable units in charge of auxiliary or technical

functions unrelated to any use case of the system (e.g. callable units belonging to the *presentation* or *persistence* layer in the example) can be ruled out. However, owing to the delocalisation and interleaving problems (Ratiu 2009) the problem and solution domain groups are not always discrete sets as each can contain units of the other. In the example, the methods *searchCustomers* and *searchProducts* in the class *BuyerController* (see Figure 2) mix problem and solution code, since they also contain code related to database access. Consequently, in many cases the only possible solution is for systems analysts to provide the information as to whether a callable unit belongs to the problem or solution domain.

4.4. Challenge 4 – process scope

Another important challenge is to establish the scope of a business process (i.e. to identify where a process instance starts and ends). While the start and end points of a business process are explicitly defined in PAISs (see Figure 3A), traditional information systems lack any explicit information about the processes supported (see Figure 3B).

Unfortunately, the information as to where a process starts and ends cannot be automatically derived from the source code. In the example illustrated in Section 3, there is not enough information to determine what methods comprise the start and end points of the business processes supported, human interaction being necessary. On the one hand, business experts know the business processes of the organisation as well as their start and end activities. On the other hand, systems analysts know what callable units in the source code support the start and end activities.

4.5. Challenge 5 – process instance scope

The lack of process-awareness in traditional information systems poses another fundamental challenge for a business process is typically not only executed in isolation, but multiple instances are executed concurrently. If a particular business activity is executed (i.e. a callable unit is invoked), this particular event has to be correctly linked to one of the running process instances. Let us consider the original business process of the example (see Figure 1). The Java application supporting it could execute the sequence '*Customer Validation*' (Customer 1), '*Customer Validation*' (Customer 2), '*Place Order*' (Customer 1) and '*Place Order*' (Customer 2) for two different customers. To obtain meaningful event logs, the activities executed by the information system must be correctly linked to either Customer 1 or Customer 2 (i.e. the customer information in this example uniquely identifies a process instance).

Correlating an activity with a data set that uniquely identifies the process instance it belongs to (e.g. the customer name) poses significant challenges. In particular, the objects to be used for uniquely identifying a process instance (i.e. what the correlation data is) must be established. If correlation objects have been identified, their location in each callable unit within the legacy source code has to be determined. The correlation data set can be explicitly defined by selecting the appropriate argument or variable in each particular callable unit. In addition, the correlation data set can be implicitly defined by choosing a classifier or data type in the source code (e.g. the Customer class in the example), so each occurrence of the selected classifier in a callable unit (e.g. a particular Customer object) is used as

correlation data. In both cases, definition of the correlation data set requires the input of business experts who know what the correlation data set is, as well as the input of systems analysts who know the location of those correlation data.

Another related problem is that the correlation data might be lost during run-time, since some callable units may not instantiate or use any object of the classifier selected as the correlation data set (e.g. *searchCustomers* or *searchProducts* in the example). For this reason, traceability mechanisms throughout callable units should be implemented to have the correlation data available at any point of the source code during system execution.

5. The solution proposed

This article proposes a technique to obtain event logs from non-process-aware systems addressing the challenges discussed above. Our proposal presents the guidelines of a generic technique, although the supporting tool, *Event Traces Injector* (ETI), is specially designed for object-oriented systems.

The technique is based on static analysis of source code combined with a dynamic analysis. *Static analysis* examines the source code in a static way, and instruments the source code by injecting code for generating specific events during its execution (cf. Section 5.1). After static analysis, the source code is *dynamically analysed* in run-time by means of the injected statements (cf. Section 5.2). Figure 4 gives an overview of the technique, the tasks carried out and the artefacts obtained (shown in grey).

5.1. Static analysis to instrument source code

The static analysis is the key stage of the technique proposed to instrument legacy source code. Source code instrumentation means that additional statements are injected into the original source code to record events during system execution. As traditional information systems lack process-awareness, this task poses several challenges (as introduced in Section 4). While challenges C1 and C2 can be addressed in a fully automated manner (Tasks 5 and 6 in Figure 4), challenges C3, C4 and C5 require input from the business expert and/or the systems analyst (Tasks 1–4 in Figure 4).

In Task 1, business experts identify the start and end business activities of the business processes under investigation. This information is necessary to deal with the

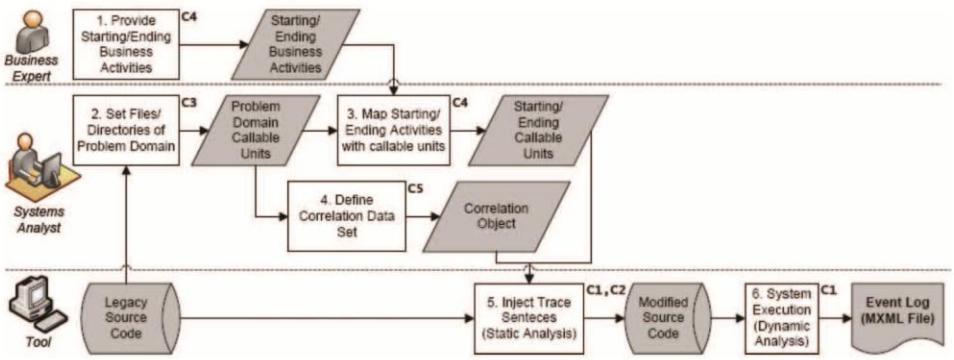


Figure 4. The overall process carried out by means of the proposed technique.

challenge of process scope (Challenge C4). In parallel, in Task 2, systems analysts examine the legacy source code and filter directories, files or sets of callable units that support business activities, (i.e. they select the callable units belonging to the problem domain). This information is necessary to reduce potential noise in the event log due to technical source code (Challenge C3). Task 3 is the mapping between start/end business activities and the callable units supporting them, which is again supported by the systems analysts (Challenge C4).

In Task 4, systems analysts establish the correlation data set that uniquely identifies a process instance (Challenge C5), selecting for this purpose a classifier (or an object kind in general for any object-oriented language). Each object instance of that classifier will be used as the correlation data set in all the callable units that use it or have it as a parameter. This information is then used during run-time when the dynamic analysis writes the event log to correlate the activities executed with the appropriate process instance. Unfortunately, some callable units could not have a correlation object in their scope, since the correlation data may not be available in some intermediate or auxiliary callable units. In order to solve this problem, the technique implements a solution based on the heuristic *principle of locality*. This solution includes callable units, whenever the correlation data are empty, in the same process instance as the last callable unit executed. This solution is implemented during the final dynamic analysis in run-time (cf. Section 5.2).

Figure 2 shows the information provided by the systems analysts for mining the 'order product' process illustrated in Section 3. The files or directories (Java files and packages in ETI, the supporting tool) that do not contain technical source code, and therefore belong to the problem domain, are marked with a tick (Task 2). The methods that support start or end activities are marked with circles, thin lines indicating start activities and thick ones end activities (Tasks 1, 3). The callable units 'validateCustomer' and 'receiveProducts' were respectively mapped with the start and end activities of the process 'Ordering Products' described in Figure 1. The *Customer* class is used as correlation data set, because it allows for uniquely identifying process instances for that particular application (Task 4). Figure 2 shows some underlined parameters (*Customer* objects) that represent the correlation information to be used in the callable unit.

After that, Task 5 consists of the syntactic analysis of the source code. The parser analyses the original source code and injects statements as required to enable event log generation. This analysis can be automated following the algorithm presented in Figure 5. During static analysis, the source code is broken down into callable units (Challenge 1), the algorithm only modifying the units belonging to the problem domain subgroup (see line 4 of Figure 5) selected by the systems analyst in Task 3. In addition, fine-grained callable units (e.g. *setter, getter, constructor, toString* and *equals* callable units) are automatically discarded (Challenge C2). Finally, one statement is injected at the beginning and one at the end of each of the callable units filtered (see lines 12 and 13 of Figure 5).

The injected statements will invoke the *WriteEvent* function at runtime, which writes the respective event into the event log (cf. Section 5.2). The injected statements (see lines 12 and 13 of Figure 5) have the following parameters:

- The name of the callable unit being executed.
- The name of the respective process, which is only known if the callable unit was mapped with one or more start or end activities.

- The correlation data defined for the callable unit, whenever the unit contains or uses an object of the classifier selected as the correlation object.
- The kind of the event, which automatically represents a start event for the first statement and an end event for the last one
- Whether the callable unit was defined by experts as a start activity
- Whether the callable unit was previously mapped with an end business activity
- The log variable, the path where the event log will be registered, which is provided by systems analysts.

Continuing the example described in Section 3, Figure 6 shows the method *addCustomer* after the injection of the statements. According to the algorithm (see Figure 5) statement S1 is added directly after the method signature. The body of the source method is then added without any changes and finally statement S2 is added after the body to the method.

5.2. *Dynamic analysis for obtaining event logs*

After static analysis, the instrumented source code can be released to production again. The new code makes it possible to write event log files according to the MXML (Mining XML) format (Günther and van der Aalst 2007), which is used by the process mining tool *ProM* (van der Aalst *et al.* 2009).

```
1  InjectTraces (CallableUnits, ProblemDomainCallableUnits, StartingCallableUnits, EndingCallableUnits)
2      InstrumentedSourceCode ← ϕ
3      For (c ∈ CallableUnits)
4          If (c ∈ ProblemDomainCallableUnits)
5              If (c ∈ StartingCallableUnits)
6                  isStartActivity ← true
7              Else If (c ∈ EndingCallableUnits)
8                  isEndActivity ← true
9              Else
10                 isStartActivity ← false
11                 isEndActivity ← false
12             statement1←WriteEvent(c.name,c.process,c.correlationSet,"start",isStartActivity,isEndActivity,log)
13             statement2←WriteEvent(c.name,c.process,c.correlationSet,"complete",isStartActivity,isEndActivity, log)
14             c'.signature ← c.signature
15             c'.body ← statement1 + c.body + statement2
16             InstrumenedtSourceCode ← InstrumentedSourceCode ∪ {c'}
17         Else
18             InstrumentedSourceCode ← InstrumentedSourceCode ∪ {c}
19     Return InstrumentedSourceCode
```

Figure 5. Algorithm to inject traces by means of static analysis.

```
1 public void addCustomer(Customer customer) {                                  signature
2   org.mxmlwriter.StaticMXMLWriter.writeEvent("addCustomer", null,
    String.valueOf(customer), "start", false, false, "C:/log.mxml");            S1
3   CustomerDAO.insert(customer);                                               body
4   org.mxmlwriter.StaticMXMLWriter.writeEvent("addCustomer", null,
    String.valueOf(customer), "complete", false, false, "C:/log.mxml");         S2
5 }
```

Figure 6. The Java method 'addCustomer' modified with the injected statements.

The MXML format represents an event log as a set of *Process* elements that contain several *ProcessInstance* elements, which each have a sequence of *AuditTrailEntry* elements (see Figure 7). Each *AuditTrailEntry* element represents an event and consists of four main elements:

- The *WorkflowModelElement* that represents the executed activity.
- The *EventType* showing whether the activity is being executed (start) or had been completed (complete)
- The *Originator* identifying the user who started or completed the activity.
- The Timestamp recording the date and time of the event.

In addition, all these elements can have a *Data* element that consists of a set of *Attributes* including related information.

When the control flow of the modified information system reaches an injected statement, a new event is added to the event log. The events are written by means of the *WriteEvent* function (see Figure 8), which is the function invoked through the injected statements, and has the parameters mentioned in the algorithm to inject statements (see Figure 5), which are established during static analysis, although the correlation data, for instance, are only known at run-time.

To add a new entry to the log file, the function starts searching for the appropriate process in the event log where the event must be written (see line 2 of Figure 8). If the process is null, then a new one is created. After that, the *WriteEvent* function examines the correlation data to determine which process instance the event has to be added to (see lines 3–5 of Figure 8). If the correlation data are empty, then the algorithm takes the correlation data of the previously executed callable unit to add the event to the correct process instance (see lines 6–9 of Figure 8). This solution is based on simple heuristics based on the principle of locality and allows for correlating events and process instances when no correlation data is available for the respective event. However, in order to add the event to a process instance, the

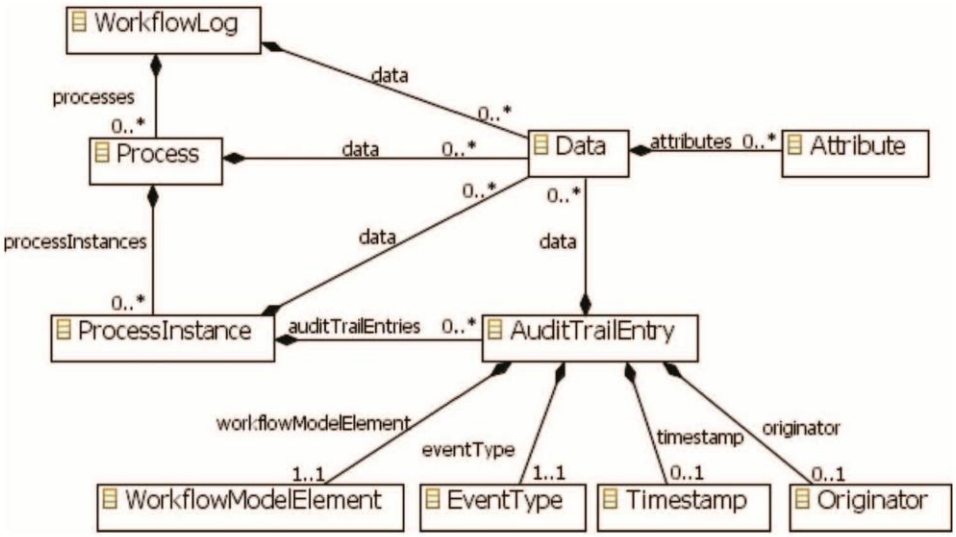


Figure 7. MXML metamodel to represent event logs.

```
1  WriteEvent (CallableUnit, ProcessName, CorrelationSet, EventType, IsStartActivity, IsEndActivity, EventLog)
2      ProcessToAppend ← EventLog.xpath ("/WorkflowLog/Process[Name='ProcessName'][last()]")
3      If (ProcessToAppend = null)
4          ProcessToAppend ← new Process (ProcessName)
5          EventLog ← EventLog ∪ {ProcessToAppend}
6      If (CorrelationSet = null)
7          CorrelationSet ← PreviousCorrelationSet
8      Else
9          PreviousCorrelationSet ← CorrelationSet
10     InstanceToAppend ← EventLog.xpath ("/WorkflowLog/Process[last()]/ProcessInstance
11         [Data/Attribute='CorrelationSet'][last()]")
12     If (IsStartActivity)
13         InstanceToAppend ← new ProcessInstance ( )
14         InstanceToAppend.Data ← InstanceToAppend.Data ∪ { CorrelationSet }
15     If (InstanceToAppend<> null)
16         auditTrailEntry = new AuditTrailEntry (CallableUnit, EventType, System.User, System.Date)
17         InstanceToAppend.Events ← InstanceToAppend.Events ∪ { auditTrailEntry }
18         ProcessToAppend.ProcessInstances ← ProcessToAppend.ProcessInstances ∪ { InstanceToAppend }
19         EventLog.Processes ← EventLog.Processes ∪ { ProcessToAppend }
20         If (IsEndActivity)
21             finishingInstance (InstanceToAppend)
```

Figure 8. Algorithm to write an event in the event log.

WriteEvent function searches for the right one taking the correlation data into account (see lines 10 and 11 of Figure 8). If the expression does not find a process instance for the correlation data (i.e. because the event belongs to a start activity), the function creates a new process instance for the correlation data.

Finally, when the *WriteEvent* function has determined the correct process instance (*InstanceToAppend* in Figure 8), it adds the event to that particular instance (see lines 12–21 of Figure 8). The event, represented as an *AuditTrailEntry* element (see Figure 7) is created in the event log model using: (i) the name of the executed callable unit that represents the *WorkflowModelElement*; (ii) the event type that is also a parameter of this function; (iii) the user logged into the operating system that executed the callable unit (or the user of the session if the system is a web application), which represents the *originator* element; and finally (iv) the system date and time when the callable unit was executed to represent the *timestamp* element.

To illustrate how the dynamic analysis stage works, Figure 9 shows a sequence diagram according to the example introduced in Section 3, while Figure 10 shows (omitting some details) a piece of the MXML event log obtained from the execution of the scenario presented in Figure 9.

In the example, when the callable unit 'validateCustomer' (message 1 in Figure 9) is invoked with the new object named 'joeDoe', a new process instance is created to record the event related to the start activity of the process. The new process instance is specially created to register the order of customer Joe Doe. After a respective process instance has been created, the callable unit 'searchProducts' is executed (message 1.1 in Figure 9). This callable unit does not have any Customer object available in its scope, so the correlation data to link the respective event to the correct process instance is missing in this case. According to the proposed heuristic-based solution, the event related to this callable unit would be recorded in the process instance related to Joe Doe, since the last used correlation data is 'joeDoe' from message 1. The following callable unit executions (messages 2–5 in Figure 9) are also included in the process instance related to Joe Doe. If, however, another

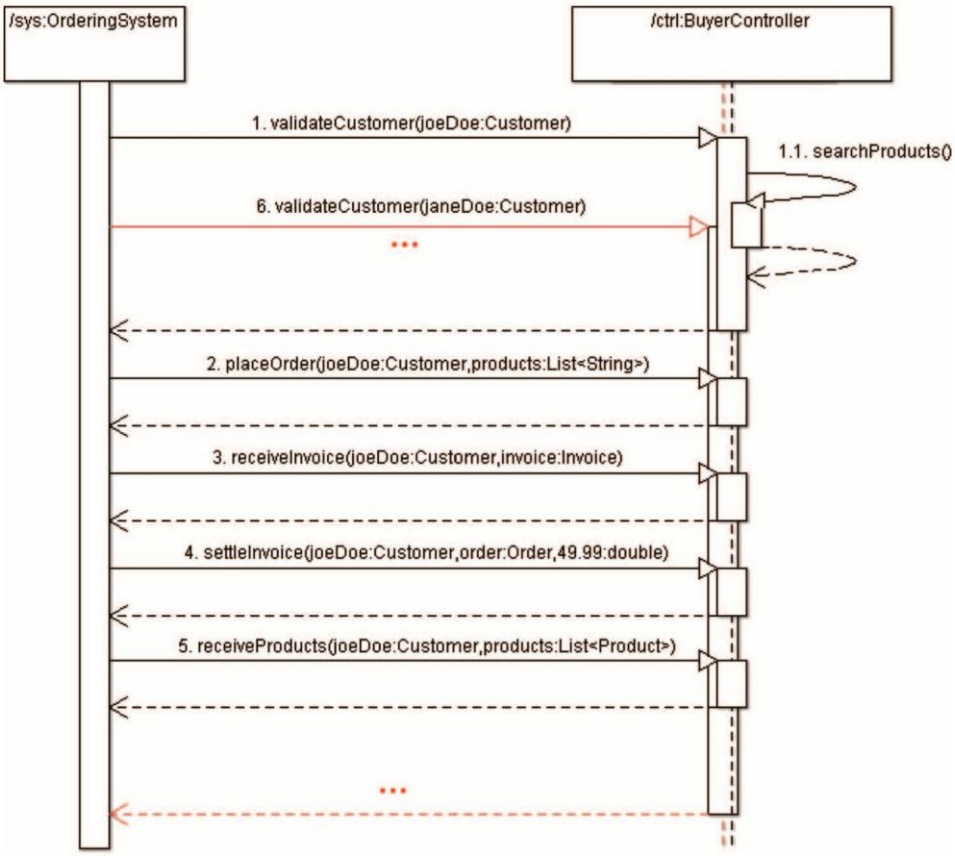


Figure 9. A sequence diagram of the product ordering application.

customer (e.g. Jane Doe) makes an order at the same time (see message 6 in Figure 9), the events created from those callable unit executions would be collected in parallel in a different process instance taking into account the Customer object for Jane Doe as correlation data.

5.3. *Final evaluation of the example*

The event log progressively obtained during the instrumented system execution is used to recover business processes. Let us imagine that the following finite set of transactions is executed with the instrumented application, which involves the registered customer *John Doe*, as well as two unregistered customers: *Jane Doe* and *Foo*.

(1) Jane Doe buys pdt1 and pdt2	(2) John Doe buys pdt1
(3) Jane Doe buys pdt3 and pdt4	(4) Foo buys pdt2, pdt3 and pdt4
(5) John Doe buys pdt4	(6) Foo buys pdt1

After the execution of those transactions, the resulting event log is analysed using *ProM* (van der Aalst *et al.* 2009) to check whether the result obtained is aligned with the original business process. The log obtained contains 1 process with 6 process

instances. It has 78 events altogether and between 12 and 16 events per process instance. Examination of the event log reveals that each process instance is related to a specific transaction executed in the application. In the next step, the genetic mining plug-in of *ProM* (Medeiros *et al.* 2007) is used for process discovery. Figure 11 shows the business process discovered using the genetic algorithm with a configuration consisting of (i) a population size of 100 and using 1000 generations in the genetic algorithm. The business process discovered has a fitness value of 0.94 in a range between 0 and 1.

The comparison between the original and the discovered business processes shows some deviations (compare Figure 1 and Figure 11). The first difference is

```
<WorkflowLog>
<Process name="Product Ordering">
      <ProcessInstance name="Joe Doe">
          <Data>
              <Attribute name="correlationData">joeDoe</Attribute>
           </Data>
          <AuditTrailEntry>
              <WorkflowModelElement>validateCustomer</WorkflowModelElement>
           </AuditTrailEntry>
          <AuditTrailEntry>
              <WorkflowModelElement>searchProducts</WorkflowModelElement>
           </AuditTrailEntry>
          <AuditTrailEntry>
              <WorkflowModelElement>placeOrder</WorkflowModelElement>
           </AuditTrailEntry>
          <AuditTrailEntry>
              <WorkflowModelElement>receiveInvoice</WorkflowModelElement>
           </AuditTrailEntry>
          <AuditTrailEntry>
              <WorkflowModelElement>settleInvoice</WorkflowModelElement>
           </AuditTrailEntry>
          <AuditTrailEntry>
              <WorkflowModelElement>receiveProducts</WorkflowModelElement>
           </AuditTrailEntry>
       </ProcessInstance>
      <ProcessInstance name="Jane Doe">
          <Data>
              <Attribute name="correlationData">janeDoe</Attribute>
          </Data>
          ...
      </ProcessInstance>
   </Process>
</WorkflowLog>
```

Figure 10. An MXML event log obtained from a particular scenario of execution.

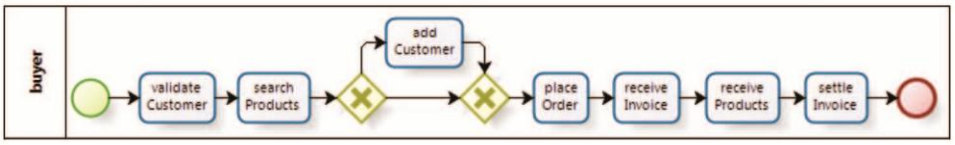


Figure 11. The business process discovered using the genetic algorithm of *ProM*.

related to the activity names, i.e. the names in the process discovered are inherited from the source code and therefore differ from the labels used in the original business process. Another important difference is that in the discovered business process, activity '*searchProducts*' (which is not present in the original process) occurs after activity '*validateCustomer*'. This deviation results from a technical method that was not filtered out by the systems analyst during static analysis. Finally, the parallel branches at the end of the process are not mined correctly (i.e. activities '*receiveInvoice*', '*receiveProducts*' and '*settleInvoice*' are carried out sequentially, instead of concurrently). This deviation is due to the fact that the operations are always executed in the same order through the application that supports the original business process. Despite these deviations, the process obtained gives a good starting point for understanding the underlying business process. In addition, the technique proposed can be applied iteratively, i.e. business experts and systems analysts can refine the information provided in order to obtain better event logs representing the business process more accurately.

5.4. Coverage of the event log

During system execution, certain coverage must be ensured in order to get an event log that considers at least an appropriate number of execution paths. The proposal considers the *function coverage*, which is commonly used in software testing (Ammann and Offutt 2008), among other research fields. This criterion counts the instrumented callable units reached or invoked during system execution and is considered for the proposal instead of others like sentence coverage, decision coverage or condition coverage, as the callable unit is the analysis unit considered according to the '*a callable unit/a business activity*' principle.

This criterion only assesses the quality of the process of obtaining the event log, i.e. the completeness of the event log regarding all the candidate activities to be added. If systems analysts provide the wrong filtering concerning the instrumented callable units (and therefore, the wrong filtering regarding the set of candidate activities to be added), then the business processes retrieved from the log might be inaccurate even if the event log covers a certain percentage of instrumented callable units.

Besides the coverage criterion, the limit to be used in each case must be established. For instance, a coverage value of 0.9 means that at least 90% of all instrumented callable units were invoked during system execution. On one hand, the coverage limit can be established at the beginning by business experts and systems analysts. On the other hand, the execution time can be previously fixed, and the coverage obtained after that time at the end of the dynamic analysis stage can be used as a significance value of the event log obtained.

Ideally, the coverage limit is a value close to 100%. However, it is unrealistic to assume that 100% coverage can be reached in practice, at least within a reasonable period of time, since coverage evolution is usually characterised by an asymptotic curve. At the beginning, when the event log is empty and no callable unit has been executed, function coverage value quickly grows. At the end, when many callable units have already been executed, the coverage percentage increases more gradually.

Even after the instrumented system is executed for a long time, a coverage value of 100% cannot be guaranteed, for a legacy system might contain unreachable code (i.e. dead code) or features rarely or never executed. For this reason, our proposal

considers a coverage limit of 60%, evaluated by considering only the instrumented callable units (not the total units). This percentage can be reached in a moderate time and provides a good estimation of how complete the event log is.

As a result of the dynamic analysis stage, event logs are obtained which can be used as the input for different process-mining algorithms. The business processes obtained using the event log obtained provide a good starting point to understand the source business process. If the results obtained do not provide sufficient quality, a new iteration of code instrumentation and dynamic analysis may be performed. Both business experts and systems analysts can refine the information provided for code instrumentation in the next iteration. As a consequence, the event log obtained during dynamic analysis might be better than the one previously obtained. In addition, experts of both kinds might stop the iterations when a sufficiently high coverage has been reached.

6. Case study

To validate the proposed technique a case study involving *CHES* (Meraner *et al.* 2009), a healthcare information system, was carried out following the formal protocol for planning, conducting and reporting case studies proposed by Brereton *et al.* (2008), thus improving the rigor and validity of the study. The following sections present in detail the main stages defined in the formal protocol: background, design, case selection, case study procedure, data collection, analysis and interpretation and validity evaluation.

ETI has been specially developed to semi-automatically support the code instrumentation in the static analysis stage. Additionally, the event logs obtained by means of the execution of the instrumented system are then dynamically analysed using *ProM* (van der Aalst *et al.* 2009). Both tools support the execution of the case study.

6.1. Background

Firstly, previous research on the topic must be identified. Related work presented in Section 2 discusses other approaches for recovering business processes from LISs and compares them with our approach (see Table 1). Our approach focuses particularly on traditional (non-process-aware) information systems and is based on static and dynamic analysis of the source code to obtain events logs. Therefore, the *object of study* is the technique proposed to obtain event logs from non-process-aware information systems, and the *purpose of this study* is the evaluation of specific properties of the proposed technique regarding its effectiveness and efficiency. Taking into account the object and purpose of the study, two main research questions are defined (see Table 2).

On the one hand, *MQ1* checks whether the technique can obtain event logs to address the challenges presented, and whether the event logs obtained allow process-mining tools to effectively discover business processes. *MQ1* determines whether the business processes are accurately recovered from the event log, i.e. if the business processes discovered represent the business behaviour of the organisation that owns the LIS. Table 2 shows two additional research questions derived from *MQ1*. The first, *AQ1*, determines whether the business processes discovered include all the elements required in the real-world business processes, while the second one, *AQ2*,

checks whether the processes mined from the event log obtained contain any elements that do not belong to the organisation's business processes. While *AQ1* is related to the completeness of the business process models, *AQ2* is more related to their specificity.

On the other hand, *MQ2* concerns the efficiency of the technique proposed for use with any information system regardless of its size. *MQ2* is also divided into two additional questions: *AQ3* assesses the time spent on the injection of the special sentences in the source code of the LIS and *AQ4* concerns the performance penalty induced by the injected statements.

6.2. Design

The study follows the *embedded case study* design according to the classification proposed by *Yin* (Yin 2003), whereby the case study consists of a single case (i.e. it focuses on a single LIS) but considers several analysis units within the case. The analysis units in this study are the different business processes supported by CHES. Therefore, the study consists in applying the technique proposed to obtain an event log, which is then mined to discover the set of business processes that is in turn analysed to answer the research questions in Table 2. For this purpose, some measures are established to answer the questions quantitatively (see Table 3).

To evaluate the effectiveness of the proposed method through question *MQ1*, the study proposes using two measures: *recall* and *precision*, commonly used in information retrieval scenarios (Raghavan *et al.* 1989), which we have adapted for

Table 2. Case study research questions.

Id	Research question
MQ1	Can the proposed technique obtain event logs from legacy information systems to effectively mine business processes?
AQ1	Can the technique obtain event logs discovering all the elements of the embedded business processes?
AQ2	Can the technique obtain event logs without discovering any elements not belonging to the business processes?
MQ2	Can the proposed technique be efficiently used with any LIS?
AQ3	How much time is spent on obtaining event logs from a LIS?
AQ4	How much time is spent on discovering business processes from event logs?

Table 3. Case study measures.

Id	Measure	Formula	Research question
M1	Recall	$R = \frac{\{\text{relevant tasks}\} \cap \{\text{recovered tasks}\}}{\{\text{relevant tasks}\}}$	AQ1
M2	Precision	$P = \frac{\{\text{relevant tasks}\} \cap \{\text{recovered tasks}\}}{\{\text{recovered tasks}\}}$	AQ2
M3	F-measure	$P = \frac{2 \cdot \text{PRECISION} \cdot \text{RECALL}}{\text{PRECISION} + \text{RECALL}}$	MQ1
M4	Event log creation time	$T_{EL} = T_{\{\text{Manual Intervention}\}} + T_{\{\text{Static Analysis}\}} + T_{\{\text{Dynamic Analysis}\}}$	AQ3
M5	Throughput time	$T_{BP} = T_{\{\text{Business Process Obtaining}\}}$	AQ4

one of business process recovery. They are used because *precision* can be seen as a measure of exactness or fidelity, whereas *recall* is a measure of completeness. On the one hand, *recall* (*M1*) represents the number of relevant elements recovered as a function of the total of relevant elements (recovered and not recovered) depicting the organisation's whole business operation. *Recall* is related to the answer to question *AQ1*, while *precision* (*M2*) represents the number of relevant elements recovered within the set of recovered elements in a business process model. An element is considered relevant if it faithfully represents business operations or the business behaviour of the organisation in the real world. *Precision* is used to answer question *AQ2*. While each business process discovered is an independent variable, these two measures are dependent variables in this study.

To evaluate these measures in a business process recovery scenario, we considered the task element as the unit element to be evaluated. Therefore, *recall* (M1) is defined as the number of true relevant tasks divided by the total number of relevant tasks, i.e. the relevant tasks of the business process and other tasks that should have been recovered but were not, while, *precision* (M2) is defined as the number of true relevant tasks divided by the total number of relevant (recovered) tasks, i.e. the sum of true relevant tasks and false relevant tasks that were incorrectly recovered.

Expert business opinion is used to evaluate these measures and ascertain whether or not a recovered task is relevant according to the organisation's current business processes. A business process is usually defined as a set of tasks or activities as well as the ordering relationships. For this reason, although *tasks* are the only element evaluated, the '*relevant task*' definition implicitly considers other business process elements related to the control flow. *Relevant tasks* are therefore defined by four conditions. The first condition specifies that the task must represent a real-life business operation within the organisation. So, the task named '*parseDate*' obtained from an auxiliary Java method does not represent any valuable business activity. This condition must not be evaluated by considering task names, since these names are inherited from legacy code and they may be biased as regards the real business activity names provided by business experts. For example, the task named '*receiveOrder*', despite its name, is not the same as 'Place Order' (see Figure 1 in Section 3) but represents the valuable business activity in charge of receiving product orders. The second condition ensures that all the tasks preceding the evaluated task must be 'recovered relevant tasks'. In order to meet this condition, there can be no non-relevant tasks with a sequence flow to the task evaluated. The third condition ensures that all subsequent tasks must be directly (or indirectly) 'recovered relevant tasks'. The second and third conditions check the control flow of a business process model focusing on recovered tasks one by one. These conditions assume that if preceding and following tasks are also relevant, then the control flow would be appropriate since the relevance of those tasks is recursively evaluated in the same manner. Finally, the fourth condition specifies that all the Data Object elements read and written by the task evaluated must also be recovered.

The measures will be evaluated according to the business process models discovered by two business experts from the organisation as well as two systems analysts belonging to the system development team. The former are familiar with the organisation's business processes and they begin by providing a sketch of the original business process supported by the system. The systems analysts, after discovering business process models from the event logs obtained, take the original business

process provided by the business experts and score the base measures like 'recovered relevant tasks', 'recovered non-relevant tasks', 'non-recovered relevant tasks', and so on. With this information, we then calculate the derived measures, precision and recall (see Table 3). Although the *precision* and *recall* measures are appropriate, there is an inverse relationship between them, so extracting conclusions to answer *MQ1* with an isolated evaluation of these measures is very difficult. For this reason, these measures are usually combined into a single measure (*M3*) known as *F-measure* (Raghavan *et al.* 1989), which consists of a weighted harmonic mean of the two.

Two other measures are also considered when answering *MQ2* and its sub-questions in relation to the efficiency of the technique under study. Measure *M4*, concerning the answer to *AQ3*, includes (see Table 3): (i) the total time spent by ETI on the injection of sentences into the legacy source code; (ii) the time spent on manual pre-intervention by both business experts and systems analysts; and finally (iii) the time spent on dynamic analysis (i.e. the time penalty when the modified source code is executed). Finally, measure *M5*, related to *AQ4*, evaluates the time spent on discovering business processes form the event log, which, in this study, is provided by *ProM*, and depends on the algorithm used for discovering them. Both *M4* and *M5* make it possible to answer the second main research question, *MQ2*.

6.3. Case selection

Brereton *et al.* (2008) state the case under study cannot be randomly selected. The case must be selected according to some criteria so that to ensure the case would provide strengthened conclusions from the execution of the case study. For this reason, the case selection stage defines a criterion list to choose a suitable case for study. Table 4 presents the five criteria established to select the most appropriate information system. *C1* is defined to ensure that the system selected is a real-life information system that is currently in the production stage and supports the business operation of an organisation or company. *C2* ensures that the system selected is a traditional information system with no built-in logging mechanism. *C3* makes it possible to effectively select a LIS that has been significantly modified. Production time is not a suitable measure for checking this criterion, since it has no direct relationship with changes to the system. Instead, *C3* considers the number of major modifications to the system that transformed the source business processes (i.e. *adaptive* and *perfective* maintenance (ISO/IEC 2006)). *C4* ensures that the system is large enough to draw representative conclusions, exceeding 75,000 lines of source code. Finally, criterion *C5* requires the system to be based on Java technology, since the parser of ETI supporting the technique at this stage is only available for Java-based systems. However, the technique proposed is not specific for

Table 4. Criteria for case selection.

Id	Criterion for case selection
C1	It must be an enterprise system
C2	It must be a non process-aware information system
C3	It must be a significantly modified legacy system
C4	It must be of a size not less than 75 KLOC
C5	It must be a Java-based system

Java-based systems, but is based on the concept of callable units, which can also be applied to other programming languages.

After evaluating several available systems according to these criteria (see Table 4), the LIS selected for study was *CHES* (*Computer-based Health Evaluation System*)[1], a healthcare information system used in several Austrian hospitals in different areas of medicine (e.g. oncology, geriatrics, psychiatry, psychosomatic medicine) for the collection, storage and graphical processing of medical and psychosocial data. In addition, *CHES* provides graphical real-time feedback of patient-related data to individualise treatment strategies and permits recording individual patient and treatment data (e.g. laboratory values, data from medical interventions, questionnaire data). Data can therefore be analysed in real-time utilising pre-defined reference and cut-off values for determining the appropriate treatment or intervention for a patient. It thus meets *C1*. *CHES* has no mechanism to register event logs, thus meeting *C2*.

The first release of *CHES* was moved to the production stage 4 years ago. During that time, the most important maintenance actions in its version history were the following:

- v0.1 was a prototype implemented in *Visual Basic*
- v1.0 was basically the same system as v0.1, but put on a more solid basis and implemented in *Java*
- In v2.0 several configuration options were implemented (e.g. customisation of questionnaires, computation of reference values)
- v3.0 involved a complete user interface redesign and added the support for creating reports
- v3.1 extended *CHES* with a new survey application, internationalisation and an update mechanism.

As a result, *CHES* underwent three major modifications and a medium-sized, its *C3* compliance thus being ensured.

The total size of *CHES* is 91.3 KLOC, ensuring *C4*. From a technological point of view, *CHES* is a *Java* application meeting *C5*, and its architecture follows the traditional structure in three layers (Eckerson 1995): (i) the *domain* layer supporting all the business entities and controllers; (ii) the *presentation* layer dealing with the user interfaces; and (iii) the *persistency* layer handling data access.

6.4. Case study procedure

In addition to the case study design, a procedure was defined to execute the study in a finite set of steps, some of which are partially executed by ETI, the tool developed to support the technique proposed. The steps defined are:

(1) After some meetings between staff from both the candidate organisation and the development team as well as the researchers, the LIS was selected according to the case selection criteria. At this point, the business experts and systems analysts who will provide the information to modify the original source code were appointed. The information provided by them is necessary to perform a later manual verification in order to evaluate measures *M1* and *M2* (see Section 6.2).

(2) The legacy source code was modified by means of ETI, which the systems analysts and researchers used according to the technique proposed (cf. Section 4), playing the roles of business experts, for which purpose they considered the first sketch of the original business processes defined by the real business experts in a meeting at the beginning of the study.
(3) Before obtaining a definitive event log, the instrumented system was executed and a test event log obtained, which was analysed using *ProM*. The accuracy of the preliminary event log was checked, as it could have had any event fault due to a non-optimal instrumentation of source code during the static analysis stage. The systems analysts were thus able to refine the information previously provided in ETI and iteratively instrument the source code (returning to the second step).
(4) When an optimal instrumentation of the legacy source code was achieved (i.e. when a first sketch of a business process could be accurately obtained after the dynamic analysis stage using the instrumented code), the instrumented LIS was released in an experimental environment. The research team then used the modified application, simulating normal users in the organisation. The execution was led by the underlying execution scenarios derived from the sketch of business processes provided by the business experts. For this case study, the instrumented LIS was executed until the *function coverage* reached a value greater than 60%.
(5) The event log obtained by applying the technique was analysed with *ProM*, which supports many techniques and algorithms to discover processes from event logs, in order to discover a set of preliminary business processes. This study used the genetic mining algorithm implemented in *ProM*, as it facilitates the discovery of the sequence flow structure. In addition, according to Medeiros *et al.* (2007) the accuracy of the genetic algorithm makes it the most suitable one.
(6) The first sketch of business processes obtained by model transformation was improved by systems analysts and researchers playing the roles of business experts. We fitted the preliminary business processes to the reality of the organisation, i.e. we added tasks that should have been recovered but were not, and removed tasks erroneously recovered. Next we assessed the accuracy of business process models by comparing each preliminary business process with its respective enhanced business process. We obtained the value of the measures proposed such as *precision* and *recall* by scoring the differences between the preliminary and enhanced business processes. Steps 2–6 were iteratively executed until business processes were obtained with an appropriate quality level.
(7) All key information related to the generation of the event log (steps 2 and 3), the business processes discovered (step 4), as well as the business expert intervention (step 5), was collected according to the data collection plan (see Subsection 6.5). The data collected in the previous step were analysed and interpreted to draw conclusions in order to answer the research questions. Subsection 6.6 presents the results obtained in this study. Finally, the case study was reported and feedback is given to the organisation and research community.

6.5. Data collection

Data to be collected and their sources were defined before the case study in order to ensure repeatability, starting with the business experts' configuration registered in

ETI to instrument the source code. Table 5 shows the last iteration of this information, i.e. the definitive details of the static analysis stage according to the iterative case study procedure (see Steps 2 and 3 in Section 6.4). It also gives the name of the processes selected by business experts together with their start and end activities, the number of domain directories, files and technical filters selected by the systems analysts and, finally, the correlation object, the *Patient* classifier, provided by experts.

The information derived from the static analysis, provided by ETI, is also collected, with Table 6 showing the last iteration, giving: (i) the number of source code files in *CHES*; (ii) the number of *Java* source code files; (iii) the number of Java files modified according to the domain files selected; (iv) the number of sentences injected; and (v) the total time needed to obtain the modified version of *CHES*.

After execution of the modified version of *CHES*, an event log was obtained, a summary of which is shown in Table 7 in its last iteration, provided by *ProM*: (i) the total processes registered; (ii) the number of process instances; (iii) the total number of events; (iv) the number of unique events; (v) the mean of events per process instance; and finally (vi) the function coverage percentage reached.

Table 8 shows the data derived by business process discovery in the last iteration (i.e. information about the business process discovered from the final, stable event log), together with the evaluation of *precision* and *recall* measures by business experts (c.f. the final business processes discovered as well as the business experts' evaluation in Appendix 1). Table 8 shows the following data for each business process:

- The number of tasks recovered (before manual intervention).
- The number of relevant tasks recovered (i.e. the number of tasks that the business experts marked as correct).

Table 5. Business expert information to perform the static analysis stage.

Process	Start activity	End activity
Patient admission	Input data patient	Register patient
Patient stay management	Select patient	Save questionnaire
Data collection management	Select questionnaire	Update patient state
Data analysis management	Check indicators	Order treatment
		Order intervention
		Print report
Number of Selected domain directories	143	
Number of Selected domain files	362	
Number of Technical filters	38	
Correlation object	*Patient*	

Table 6. Static analysis results.

Feature	Value
LOC	91266
Number of Java files	822
Number of processed Java files	181
Number of instrumented callable units	336
Static analysis time	8′31″ (511435 ms)

Table 7. Event log analysis.

Feature	Value
Number of processes	4
Number of process instances	336
Number of events	1185
Number of events classes	204
Mean of events per instance	3.53
Function coverage	60.7%

- The number of non-relevant tasks recovered (i.e. tasks removed from the business process as they do not represent a business activity).
- The number of non-recovered relevant tasks (i.e. relevant tasks added by experts to the business processes because they were not recovered).
- The *recall* values for each final business process.
- The *precision* values for each final business process.
- The harmonic mean between both measures.
- The fitness value (between 0 and 1) obtained by executing the genetic algorithm in *ProM*, which represents the accuracy level of the transformation from event log to business processes.
- The time spent on discovering business processes with *ProM* (i.e. the *M5* measure).

Finally, Table 9 contains similar information regarding all the previous generations of event logs up to the final one, focusing on the different time values spent on each stage of the proposed technique: (i) manual intervention time; (ii) static analysis time; (iii) dynamic analysis time; (iv) event log obtaining time (i.e. the sum of the three previous values according to *M4*); and (v) business process discovery time (*M5*). In addition, Table 9 summarises the remaining measures like (vi) recall, *M1*; (vii) precision, *M2*; (viii) F-measure, *M3*; and finally (ix) the fitness value obtained during the process discovery algorithm.

6.6. *Analysis and interpretation*

After data collection, data analysis obtained the evidence chains to answer the research questions. Four iterations were carried out until an appropriate event log was obtained, which had a total of 1185 events with an average of seven events per process instance (see Table 7). It comprises four processes in total, related to the four processes at the beginning of the static analysis stage (see Table 5). After this, a business process was discovered using *ProM*, with 21 recovered tasks, of which 11 were relevant and 10 non-relevant, 7 additional relevant tasks then being added by the business experts (c.f. the final discovered business processes and business experts' evaluation in Appendix 1).

6.6.1. *Effectiveness evaluation*

In answer to research question *MQ1*, the values of the *recall* and *precision* measures were 0.638 and 0.633, respectively (the average for the four business processes). As a

Table 8. Business process model description.

Business process model name	Number of recovered tasks (RcT)	Number of recovered relevant tasks (RcRvT)	Number of recovered non-relevant tasks (RcNRvT)	Number of non-recovered relevant tasks (NRcRvT)	Recall (RcRvT/ RcRvT + NRcRvT)	Precision (RcRvT/ RcRvT + RcNRvT)	F-measure (2 × P × R)/ (P + R)	Fitness value	Business process time (s)
Patient admission	5	2	3	3	0.400	0.400	0.400	0.928	46
Data collection management	2	2	0	3	0.400	1.000	0.571	0.975	5
Data analysis management	5	4	1	0	1.000	0.800	0.889	0.928	91
Patient stay management	9	3	6	1	0.750	0.333	0.462	0.886	145
Total	21	11	10	7	–	–	–	–	287
Mean	5.3	2.8	2.5	1.8	0.638	0.633	0.635	0.929	72
Standard deviation	2.9	1.0	2.6	1.5	0.29	0.32	0.22	0.04	60

Table 9. Time and measure values in iterations concerning event log generations.

Iteration	Manual intervention time (min)	Static analysis time (min)	Dynamic analysis time (min)	Event log obtaining time (min) [M4]	Throughput time (min) [M5]	Recall [M1]	Precision [M2]	F-measure [M3]	Fitness value
1	90	9,8	$K \rightarrow 0$	253	71	0.833	0.357	0.500	0.708
2	30	9,4	$K \rightarrow 0$	186	19	0.778	0.583	0.667	0.930
3	15	9,0	$K \rightarrow 0$	164	7	0.500	0.600	0.545	0.889
4	5	8,5	$K \rightarrow 0$	147	5	0.638	0.633	0.635	0.929
Total	140	37	$K \rightarrow 0$	751	102	–	–	–	–

consequence, the *F-measure* value was 0.635. These values obtained for the last iteration were more appropriate than those obtained in previous iterations (see Figure 12).

Figure 12 shows the evolution of all these measures for the previous versions of event logs. In the first iteration the *recall* and *precision* measures were respectively 0.833 and 0.357. On the one hand, a higher *recall* value means that the proposed method recovers a higher number of relevant business elements. On the other hand, a lower *precision* value contrasts with a high *recall* value, which means that a large number of non-relevant tasks have been recovered, non-relevant in most cases because their respective callable units (from which were obtained) do not represent business activities, but, for example, technical source code (e.g. auxiliary methods). In some cases the tasks may have been classified as non-relevant owing to their inappropriate level of granularity (e.g. their business knowledge is already included in other larger relevant tasks).

The precision and recall values obtained for the first iteration were not appropriate. In fact, the first sketch of business processes obtained from the first event log had several mistakes and some deviations from the real-world processes (see Appendix 2). In any event, the results obtained for the first event log were to be expected since there is an inverse relationship between *precision* and *recall* measures. Indeed, the *precision* value should, ideally, always be 1.0 for any *recall* value but, according to Davis and Goadrich (2006), this is not possible in practice. Therefore, due to this inverse relationship, it is possible to increase one at the cost of reducing the other. This means that the proposed method could reduce its *recall* value by recovering fewer tasks, at the cost of reducing the number of non-relevant tasks recovered, i.e. increasing the *precision* value. These results were progressively obtained in subsequent iterations (see Figure 12). The above result obtained in the fourth and last iteration was the best, since the *recall* and *precision* values were more balanced. Indeed, the *F-measure* can be considered as a special case of the general F_α*-measure* (1), where α is 1. The α value is used to weight *precision* in relation to *recall*, and the selected F_1*-measure* (*M3* in Table 3) thus signifies that both *precision* and *recall* are equally important. In this study, the F_1*-measure* had an average value of 0.63.

$$F_\alpha = \frac{(1+\alpha)\cdot \text{PRECISION}\cdot \text{RECALL}}{\alpha\cdot \text{PRECISION} + \text{RECALL}} \tag{1}$$

In order to answer *AQ1* and *AQ2*, and therefore *MQ1*, the values obtained were additionally compared with reference values from other experiments with model recovery in the literature (Ye and Fischer 2002, Garcia *et al.* 2006, Lucrédio *et al.* 2008). We found reports of *recall* and *precision* values close to 60%, our benchmark values. The values obtained for our measures (*precision* = 64%, *recall* = 63% and F_1*-measure* = 63%) were therefore above the reference value. Thereby, sub-questions *AQ1* and *AQ2* could be answered positively, and *MQ1* (see Table 2) was evaluated as true, i.e. the technique proposed could obtain event logs from legacy information systems to effectively mine business processes.

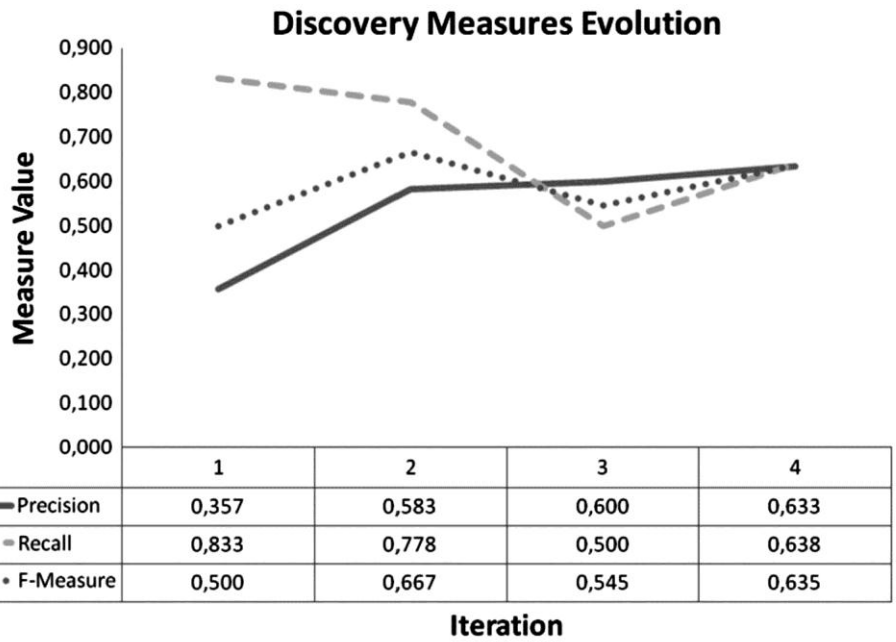
6.6.2. *Efficiency evaluation*

In order to answer *MQ2*, sub-questions *AQ3* and *AQ4* had also to be answered. The time spent obtaining the event log (measure *M4*) consisted of the manual intervention time, static analysis time and dynamic analysis time. Firstly, $T_{\{\text{Manual}}$

$_{\text{Intervention}\}}$ was only 5 min, since at that time it was only necessary to accomplish small changes using ETI, the tool developed. However, if we consider the time values of previous iterations (see Table 9), the total time concerning manual intervention was $T_{\{\text{Manual Intervention}\}} = 140$ min. Thereby, $T_{\{\text{Manual Intervention}\}}$ represents the greatest part of the total time taken to obtain an event log. However, Figure 13 shows how $T_{\{\text{Manual Intervention}\}}$ drastically decreased in subsequent iterations.

Secondly, $T_{\{\text{Static Analysis}\}}$ was 8.5 min for the last iteration, and the time values for all iterations were quite similar (between 8.5 and 9.8 min). These values did not decrease because the time spent on the static analysis of the legacy source code depended on the number of lines of source code, which was constant for all iterations. However, the minimal decrease in later iterations (see Figure 13) was due to the fact that the problem domain was slightly reduced by discarding some additional technical source code parts. As a consequence, less time was spent injecting the trace statements.

Thirdly, $T_{\{\text{Dynamic Analysis}\}}$ is considered a tiny performance penalty owing to the execution of injected statements. Indeed, the ratio of injected statements per line of source code was only 0.0074 (see Table 6), i.e. the trace statements represent only 0.7% of the entire system. In addition, this penalty is constant with respect to the system size, since it only affects the response time of each system's service or functionality. For this reason $T_{\{\text{Dynamic Analysis}\}}$ can be considered negligible. Summarising all these time values, the event log creation time (the *M4* measure) was 13.5 min for the last iteration, and 177 min in total for all iterations, which seems a reasonable time to transform a traditional LIS with 91.3 KLOC into an information system incorporating a mechanism to appropriately register event logs. Therefore, question *AQ3* was answered positively.



	1	2	3	4
Precision	0,357	0,583	0,600	0,633
Recall	0,833	0,778	0,500	0,638
F-Measure	0,500	0,667	0,545	0,635

Figure 12. Recall, precision and F-measure evolution during previous iterations.

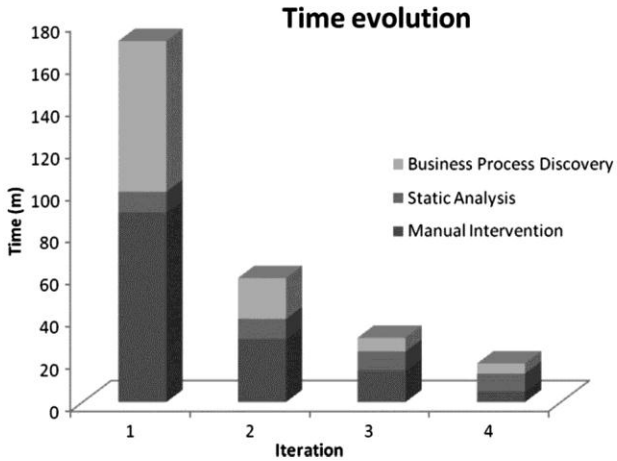


Figure 13. Time scale evolution during previous iterations.

In answer to sub-question *AQ4* the time spent on discovering business process, $T_{\{\text{Business Process Obtaining}\}}$ was 5 min for the last iteration. This time progressively decreased over the four iterations (see Figure 13), event logs gradually becoming more agglutinated and concise. The time (5 min) seemed feasible for an event log with 1185 events and obtained from the execution of the instrumented *CHES* system until it reached a *function coverage* of above 60% regarding the total of instrumented callable units. This coverage was achieved when the total event classes in the log was 204 (see Table 7), which represents 60.7% of 336 instrumented callable units (see Table 6). In addition, the genetic algorithm obtained business processes with a fitness value of 0.93, high as it could be between 0 and 1. Therefore, question *AQ4* was answered as true. Consequently, so was *MQ2*, which means that our technique can be efficiently used to discover business processes from event logs obtained.

6.7. Threats to validity

Finally, the validity of the results had to be assessed as unbiased and *true* for the whole population for which we wish to generalise the results. This section presents the threats to the validity of this case study and possible actions to mitigate them. There are mainly three types of validity: internal, construct and external.

A major threat to the *internal validity* concerned ETI, the tool developed to modify the source code, and *ProM*, used to discover business processes. These specific tools affected the results since the values measured might have been different if the business processes had been obtained with another tool. To mitigate this threat, the study could be replicated using different tools and comparing the results, or different mining algorithms of *ProM* could be used.

Regarding the *construct validity*, the measures proposed were appropriate for measuring the variables and answering the research questions. The *precision* and *recall* measures allowed us to check whether the business processes obtained accurately represented the organisation's business behaviour. Nevertheless, the reference value of around 60% taken from the literature (c.f. Section 6.6.1) to

compare the results obtained may be rather relative. Unfortunately, there are not enough benchmark values for these metrics in the process-mining field, so this threat is not easy to mitigate. On the other hand, our approach depends on accurate logs. To partially mitigate this risk, the proposed technique considers the function coverage criteria. Another threat to the construct validity concerns the execution of the instrumented system, since its use was simulated, not used in a production environment. To address this threat, we plan to move an instrumented version of *CHES* to the production stage in order to collect larger event logs, from real life.

Finally, *external validity* is concerned with the generalisation of the results. This study considers traditional LISs as the whole population. Thus, the outgoing results could be generalised to this population. However, the specific platform of the selected case is a threat that should be noted. Thus, the results are strictly extended to those legacy and non-process-aware information systems based on *Java* language. However, the results expected for other kinds of object-oriented systems could be similar to the results of this case study. In order to mitigate this threat, the study should be replicated using information systems based on different platforms, and the results obtained in these studies compared for a better understanding of the performance of the proposed technique.

7. Conclusions and future work

Business knowledge preservation is an important challenge to be addressed in every software modernisation project in order to align new modernised information systems with the real-world business processes of an organisation. Business process mining offers a promising outlook to deal with that challenge. There are many techniques for discovering business processes from event logs (Günther and van der Aalst 2007), which are automatically obtained through execution of PAISs. However, traditional LISs usually have no inherent mechanism to create event logs, so all efforts made in the process mining research field are inapplicable for information systems of this kind. To solve this problem, this article presents a novel technique based on the static and dynamic analysis of source code to obtain event logs from non-process-aware systems. The event log thus obtained can be used to discover business processes in the same way as an event log obtained from any PAIS. This is a very ambitious goal, as at least five key challenges must be addressed: (i) missing process-awareness, (ii) granularity, (iii) discarding technical code, (iv) process scope and (v) process instance scope.

The technique is divided into two stages. Firstly, the proposed technique applies static analysis for injecting special statements into the source code. This static pre-analysis instruments the original legacy source code by adding event logging capability. The second stage consists of dynamic analysis performed during execution of the instrumented system, which progressively creates an event log. The event logs obtained are then used to discover business processes using process mining techniques.

In a typical scenario, the static analysis of a LIS has to be performed once, and then the modified source code can be dynamically analysed several times to obtain event logs. However, the feedback obtained by business experts and systems analysts after the first static and dynamic analyses can be used to iteratively and incrementally refine subsequent instrumentations of source code. As a consequence,

the accuracy levels of the business processes discovered from event logs increase from iteration to iteration.

The proposed technique has been validated by means of a case study involving a real-life LIS which is running in several Austrian hospitals. The case study was conducted following a formal protocol. Its results show that the technique proposed is able to obtain event logs to effectively and efficiently discover the business processes implicitly supported by the healthcare system. We argue that our approach works effectively, because it facilitates obtaining business processes with high accuracy levels, (i.e. the business processes discovered faithfully represent what actually happens in those hospitals). Furthermore, we claim that business processes can be identified efficiently since it operates in a reasonable time. As the proposed technique includes the capturing of event logs, it makes it possible for all the research and development efforts carried out in the process mining field to be exploited for traditional LISs.

Our future work will focus on adding new features to the proposed technique, such as a traceability mechanism to take into account the call hierarchies to deal with lost and scattered correlation data. In addition, the case study will be replicated involving additional LISs.

Acknowledgement

This work was supported by the Spanish FPU Programme, and by the R + D projects ALTAMIRA (JCCM, PII2I09–0106–2463), PEGASO/MAGO (MICIN and FEDER, TIN2009–13718-C02–01), MOTERO (JCCM and FEDER, PEII11–0366–9449) and MEDUSAS (CDTI (MICINN), IDI–20090557). It received additional support from the Quality Engineering group at the University of Innsbruck, Austria. Finally, the case study was carried out with the invaluable help of Bernhard Holzner and Gerhard Rumpold of the Innsbruck Medical University.

Note

1. http://www.ches.at

References

Ammann, P. and Offutt, J., 2008. *Introduction to software testing*. New York, NY: Cambridge University Press.

Brereton, P., *et al.*, 2008. Using a protocol template for case study planning. *In*: *Evaluation and Assessment in Software Engineering (EASE'08)*. Bari, Italia: 1–8.

Cai, Z., Yang, X., and Wang, W., 2009. Business process recovery for system maintenance – an empirical approach. *In*: *25th International Conference on Software Maintenance (ICSM'09)*, 20–26 September, Edmonton, Canada. Edmonton, Canada: IEEE CS, 399–402.

Castellanos, M., *et al.*, 2009. Business process intelligence. *In:* J.J. Cardoso and W.M.P. van der Aalst, eds. *Handbook of research on business process modeling*. Hershey, PA: Idea Group Inc., 456–480.

Cornelissen, B., *et al.*, 2009. A systematic survey of program comprehension through dynamic analysis. *IEEE Transactions On Software Engineering*, 35 (5), 684–702.

Davis, J. and Goadrich, M., 2006. The relationship between precision-recall and ROC curves. *In: Proceedings of the 23rd International Conference on machine learning*, 25–29 June, Pittsburgh, Pennsylvania. Madison, Wisconsin, USA: ACM, 233–240.

Di Francescomarino, C., Marchetto, A., and Tonella, P., 2009. Reverse engineering of business processes exposed as web applications. *In: 13th European Conference on Software Maintenance and Reengineering (CSMR'09)*, 24–27 March, Fraunhofer IESE, Kaiserslautern, Germany. Washington, DC: IEEE Computer Society, 139–148.

Dumas, M., van der Aalst, W., and Ter Hofstede, A., 2005. *Process-aware information systems: bridging people and software through process technology*. Hoboken, NJ: John Wiley & Sons.

Eckerson, W., 1995. Three tier client/server architecture: achieving scalability, performance and efficiency in client server applications. *Open Information Systems*, 10 (1), 3.

Eisenbarth, T., Koschke, R., and Simon, D., 2003. Locating features in source code. *IEEE Transactions on Software Engineering*, 29 (3), 210–224.

Funk, M., *et al.*, 2010. In situ evaluation of recommender systems: framework and instrumentation. *International Journal of Human–Computer Studies*, 68 (8), 525–547.

Garcia, V.C., *et al.*, 2006. From specification to experimentation: a software component search engine architecture. *In: 9th International Symposium on Component-Based Software Engineering (CBSE 2006)*, 29 June–1 July, Västerås, Sweden. Berlin, Germany: Springer-Verlag. LNCS 4063, 82–97.

Ghose, A., Koliadis, G., and Chueng, A., 2007. Process discovery from model and text artefacts. *In: IEEE Congress on Services (Services'07)*, 9–13 July, Salt Lake City, UT. Available from: http://ro.uow.edu.au/cgi/viewcontent.cgi?article=1573&context=info papers, 167–174.

Günther, C.W., Rozinat, A., and Aalst, W.M.P., 2010. Activity mining by global trace segmentation. *In:* S. Rinderle-Ma, S. Sadiq, and F. Leymann, eds. *Business process management workshops*. Berlin, Heidelberg: Springer, 128–139.

Günther, C.W. and van der Aalst, W.M.P., 2007. A generic import framework for process event logs. Business Process Intelligence Workshop (BPI'06). *Lecture Notes in Computer Science*, 4103, 81–92.

Heuvel, W.-J.v.d., 2006. *Aligning modern business processes and legacy systems: a component-based perspective (Cooperative information systems)*. Cambridge, MA: The MIT Press.

Hofstra, P., 2009. *Analysing the effect of consumer knowledge on product usability using process mining techniques*. Thesis (Masters). Eindhoven, The Netherlands, Technische Universiteit Eindhoven, 71.

Ingvaldsen, J.E. and Gulla, J.A., 2008. Preprocessing support for large scale process mining of SAP transactions. Business Process Intelligence Workshop (BPI'07). *Lecture Notes in Computer Science*, 4928, 30–41.

ISO/IEC, 2006. ISO/IEC 14764:2006. *Software engineering – Software life cycle processes – Maintenance* [online]. ISO/IEC. Available from: http://www.iso.org/iso/catalogue_detail.htm?csnumber=39064 [Accessed 24 May 2011].

Lucrédio, D., Fortes, De M., and Whittle, J., 2008. MOOGLE: a model search engine. *In*: K. Czarnecki, I. Ober, J.-M. Bruel, A. Uhl, and M. Völter, eds. *Model driven engineering languages and systems*. Berlin/Heidelberg: Springer, 296–310.

Medeiros, A.K., Weijters, A.J., and Aalst, W.M., 2007. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14 (2), 245–304.

Mens, T., 2008. *Introduction and roadmap: history and challenges of software evolution*. Vol. 1. Springer Berlin Heidelberg: Software Evolution, 1–11.

Meraner, V., *et al.*, 2009. Development of a screening tool for the identification of psychooncological treatment need in breast cancer patients. *Psycho-Oncology*, 18 (9), 974–983.

Motahari, H.R., *et al.*, 2007. *Process spaceship: discovering process views in process spaces*. Berlin Heidelberg, Germany: The University of New South Wales, Australia.

Motahari Nezhad, H., *et al.*, 2008. Exploration of discovered process views in process spaceship. *In:* A. Bouguettaya, I. Krueger and T. Margaria, eds. *Service-oriented computing – ICSOC 2008*. Vol. 5364. Berlin/Heidelberg: Springer, 724–725.

Paradauskas, B. and Laurikaitis, A., 2006. Business knowledge extraction from legacy information systems. *Journal of Information Technology and Control*, 35 (3), 214–221.

Pérez-Castillo, R., *et al.*, 2009. MARBLE: a modernization approach for recovering business processes from legacy systems. *In: International Workshop on Reverse Engineering Models from Software Artifacts (REM'09)*, Lille, France, Simula Research Laboratory Reports. Available from: http://portal.acm.org/citation.cfm?id=1686245, 17–20.

Raghavan, V., Bollmann, P., and Jung, G. S., 1989. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems*, 7 (3), 205–229.

Ratiu, D., 2009. Reverse engineering domain models from source code. In: *International Workshop on Reverse Engineering Models from Software Artifacts (REM'09)*, 13–16 October, Lille, France. Washington, DC: Simula Research Laboratory, 13–16.

Rozinat, A., *et al.*, 2009. Process mining applied to the test process of wafer scanners in ASML. *Transactions on Systems Man and Cybernetics–Part C*, 39 (4), 474–479.

Rozman, I., Györkös, J., and Dogsa, T., 1991. Relation between source code metrics and structure analysis metrics. *In*: *Proceedings of the 3rd European Software Engineering Conference*. London, UK: Springer-Verlag, 332–342.

Serrour, B., *et al.*, 2008. Message correlation and business protocol discovery in service interaction logs. *In:* Z. Bellahsène and M. Léonard, eds. *Advanced information systems engineering*. Vol. 5074. Berlin/Heidelberg: Springer, 405–419.

Trkman, P., *et al.*, 2010. The impact of business analytics on supply chain performance. *Decision Support Systems*, 49 (3), 318–327.

Ulrich, W.M., 2002. *Legacy systems: transformation strategies*. Saddle River, NJ: Prentice Hall.

Ulrich, W.M. and Newcomb, P.H., 2010. *Information systems transformation. Architecture driven modernization case studies*. Burlington, MA: Morgan Kauffman.

van der Aalst, W. and Weijters, A.J.M.M., 2005. Process mining. *In:* M. Dumas, W. van der Aalst, and A. Ter Hofstede, eds. *Process-aware information systems: bridging people and software through process technology*. John Wiley & Sons, 235–255.

van der Aalst, W.M.P., *et al.*, 2009. ProM: the process mining toolkit. *In*: *7th International Conference on Business Process Management (BPM'09) – Demonstration Track*. Vol. 489. Ulm, Germany, Berlin, Germany: Springer-Verlag. 1–4.

Weske, M., 2007. *Business process management: concepts, languages, architectures*. Leipzig, Alemania: Springer-Verlag Berlin Heidelberg.

Ye, Y. and Fischer, G., 2002. Supporting reuse by delivering task-relevant and personalized information. *In*: *24th International Conference on Software Engineering*, 19–25 May, Orlando, Florida. New York, NY: ACM, 513–523.

Yin, R.K. 2003. *Case study research. Design and methods*. 3rd ed. London: Sage.

Zou, Y., and Hung, M., 2006. An approach for extracting workflows from e-commerce applications. *In*: *Proceedings of the Fourteenth International Conference on Program Comprehension*. Washington, DC: IEEE Computer Society, 127–136.

Appendix 1. Business experts' evaluation

This appendix shows the business process discovered from the *CHES* system using our technique. Figure A1 shows the assessment made by systems analysts and researchers who play the role of business experts in the case study. The assessment is differentiated with colours: (i) green tasks with bold type represent relevant tasks recovered (i.e. tasks approved by experts); (ii) red tasks with bold type represent non-relevant tasks recovered (i.e. tasks removed by experts); and finally (iii) grey tasks with italic represent non-recovered relevant tasks (i.e. tasks added by experts).

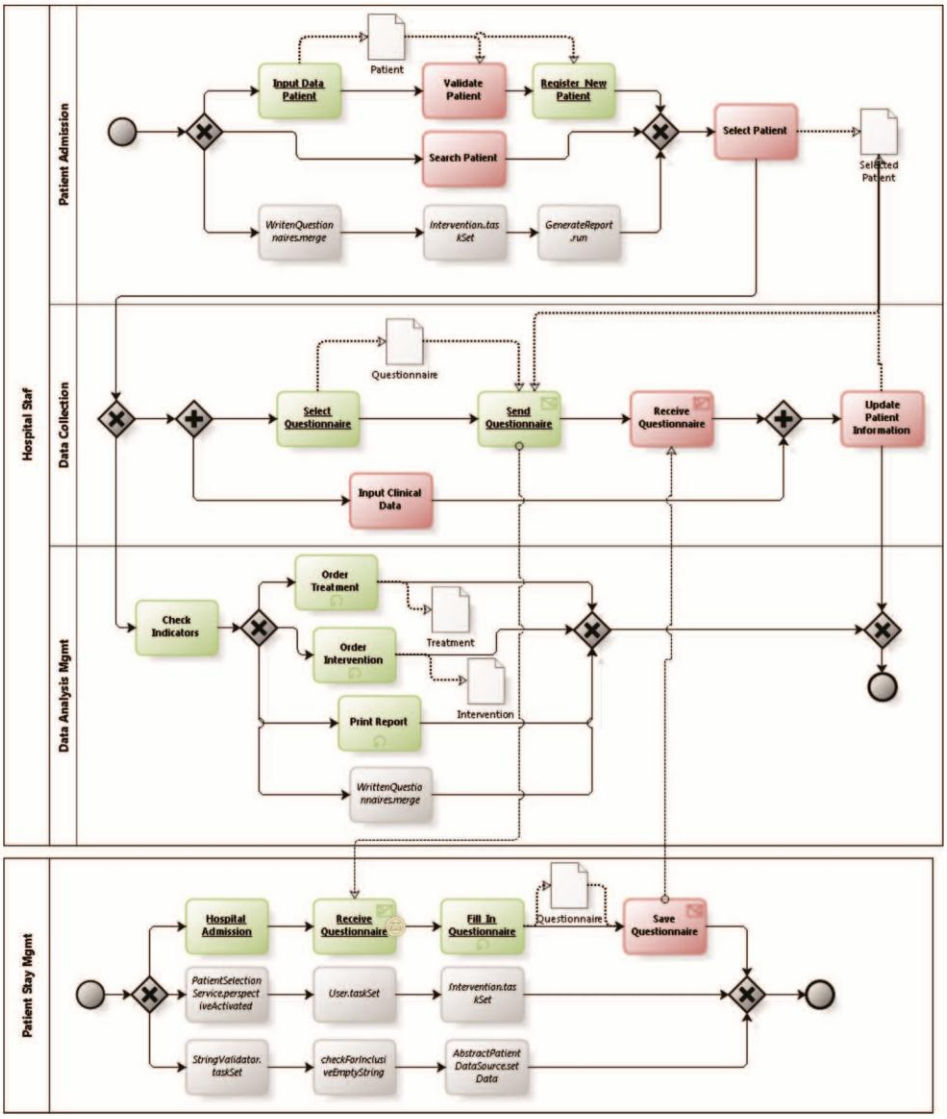


Figure A1. The CHES business process after the final assessment.

Appendix 2. The first sketch of a business process

Figure A2 shows the first sketch of business process discovered from *CHES* system using the proposed technique. The business process depicted, which corresponds to the first iteration, has a spaghetti-like control flow and some isolated sub-processes and is too fine-grained. As a consequence, the precision and recall values obtained for this business processes were inappropriate (cf. Section 6.6).

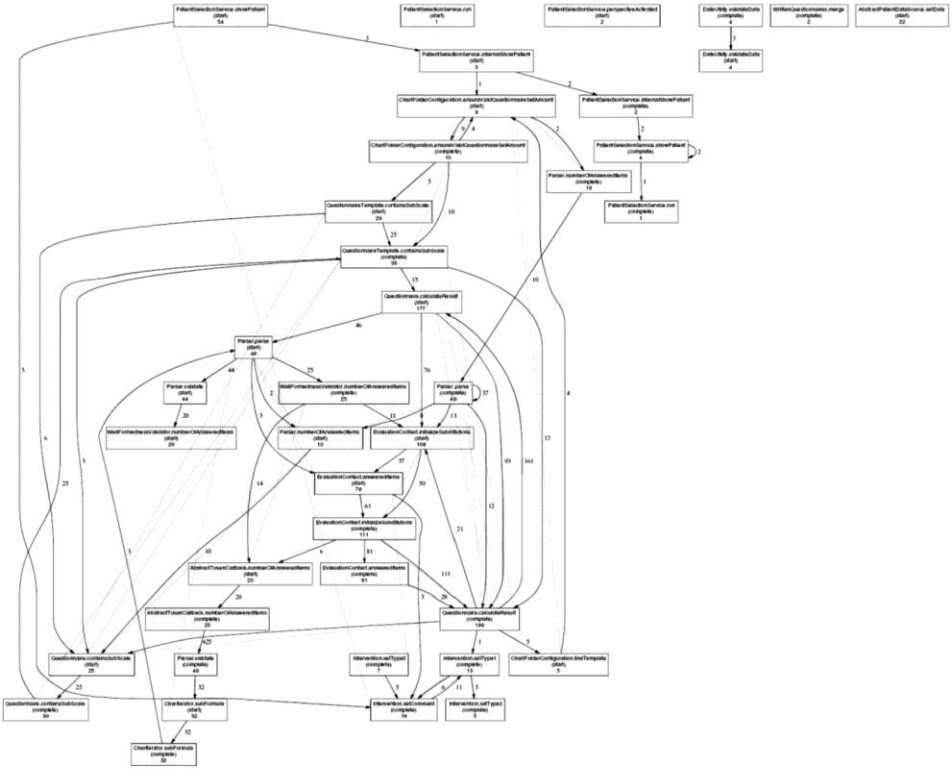


Figure A2. A first sketch of the business process supported by *CHES*.

CHAPTER 4

Supporting Tool

Due to the extensible nature of MARBLE, it is now online available as an Eclipse plugin.

4. SUPPORTING TOOL

This chapter provides relevant implementation details about the tool developed to support the static technique presented in Section 3.2. This tool has been released as an Eclipse™ application. Its future extensibility and integration with other related tools is consequently ensured (e.g., an Eclipse™ plug-in to support the dynamic technique or any other MARBLE-based technique).

This chapter consists of a paper published in the proceedings of the 27th IEEE International Conference on Software Maintenance. This conference is considered as CORE A (the top quality level) according to the ERA Conference Ranking. The reference of this paper is: *Pérez-Castillo, R., M. Fernández-Ropero, I. García Rodríguez de Guzmán, and M. Piattini, MARBLE. A Business Process Archeology Tool, in 27th IEEE International Conference on Software Maintenance (ICSM'11). 2011, IEEE Computer Society: Williamsburg, Virginia, USA. p. 578-581.*

2011 27th IEEE International Conference on Software Maintenance (ICSM)

MARBLE. A Business Process Archeology Tool

Ricardo Pérez-Castillo, María Fernández-Ropero, Ignacio García-Rodríguez de Guzmán and Mario Piattini
Alarcos Research Group, University of Castilla-La Mancha
Paseo de la Universidad 4 13071, Ciudad Real, Spain
{ricardo.pdelcastillo, marias.fernandez, ignacio.grodriguez, mario.piattini}@uclm.es

***Abstract*—Modernization of legacy information systems is usually triggered by the need of introducing new business requirements or due to the technology obsolescence. During modernization software projects, there exists a lot of business knowledge that was embedded in source code owing to progressive maintenance, which is not present anywhere else. In order to preserve embedded business knowledge this paper presents MARBLE, a tool to recover business processes from legacy information systems. MARBLE provides an extensible framework, thus it is developed as an Eclipse™ plugin to ensure its future extension. So far, MARBLE supports Java-based system and focuses mainly on legacy source code. To facilitate its adoption in the industry, MARBLE has been applied in some real-life modernization projects, which demonstrated that MARBLE is suitable to retrieve business processes and facilitated its continuous improvement to reach an appropriate maturity level.**

Keywords—Reverse Engineering; Business Process; Model-Driven Development; Knowledge Discovery Metamodel

I. INTRODUCTION

Reverse engineering techniques have become very important within the maintenance process providing several benefits. Firstly, reverse engineering allows maintainers to retrieve abstract representations to facilitate the comprehension of different legacy systems. For example, it focuses on relational databases [6], aspect oriented systems [3], quality of the system design [8], links between e-mail and source code [2], and so on. Secondly, abstract representations obtained by reverse engineering from legacy systems can be refactored to improve their maintainability or add new functionalities to evolve legacy systems.

To address the mentioned maintenance activities, reverse engineering techniques are nowadays well-supported by tools which often obtain artifacts at system design abstraction level (e.g., class or sequence diagrams from source code) [5]. However, software engineering industry is demanding additional reverse engineering techniques and tools to retrieve business-aware artifacts at higher abstraction level [11]. In fact, the Software Engineering Institute (SEI) argues that business rules recovery is the cornerstone to evolutionary maintenance towards modern paradigms like Service Oriented Architecture (SOA) [12].

To meet these demands, business process archeology has emerged as a set of techniques and tools to recover business processes from source code [15]. Maintenance benefits of business process archeology are that they preserve business behavior buried in legacy source code and it retrieves business processes providing more opportunities for refactoring (due to the higher abstraction level).

There are some techniques in literature that support business process recovery. For example *Zou et al.* [19] recover workflows by statically analyzing source code and applying some heuristic rules to discover business knowledge. *Paradauskas et al.* [14] retrieve business knowledge through the inspection of the data stored in databases. These studies rely solely on static analysis. Thereby, other solutions have been suggested based on dynamic analysis. For example, *Di Francescomarino et al.* [7] consider graphical user interfaces of web applications to discover business processes. *Cai et al.* [4] combine requirement reacquisition based on use cases with dynamic and static analysis techniques. Finally, *Van der Aalst et al.* [18] focus on mining business processes from event logs registered during system execution.

This paper particularly presents MARBLE™, a technique and tool, supporting business process archeology [15]. MARBLE™ is an extensible framework, although it so far supports static analysis of legacy source code in a similar way than technique proposed by *Zou et al.* However, MARBLE™ applies a set of business patterns formalized by model transformations. The main difference regarding mentioned proposals is that MARBLE™ is easily automatable and highly extensible due to it follows model-driven architecture principles (i.e., it considers all involved software artifacts as models, and it provides formal transformations to move models between different abstraction levels).

MARBLE™ tool [1] is implemented as an *Eclipse*™ plugin improving its extension and integration with other techniques and tools as well as to facilitate its adoption by the industry. The tool creates and manages an entire repository by integrating code models from legacy information systems. After that, the tool allows maintainers to discover, visualize and edit business process models.

The effectiveness and suitability of the tool has been demonstrated through several industrial case studies which the tool has been applied to a healthcare system, an e-government system, enterprise systems, among other kinds of legacy systems.

The remaining of the paper is organized as follows: Section II explains MARBLE™ technique supported by the tool. Section III presents in detail functionalities and implementation details of MARBLE™ tool. Section IV briefly summarizes the case studies which the tool has been used. Finally, Section IV discusses the conclusions of this work.

978-1-4577-0664-6/11/$26.00 ©2011 IEEE

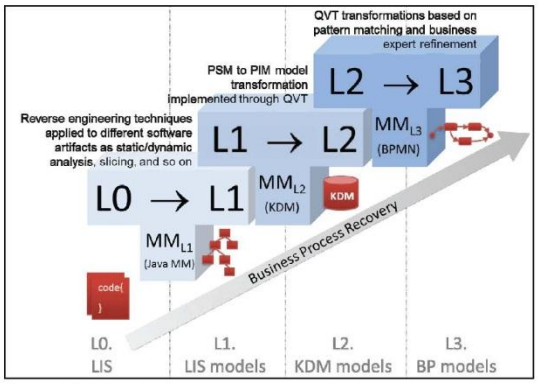


Figure 1. MARBLE™, the technique to support business process archeology

II. Business Process Archaelogy Procedure

Business process archeology [15] studies the business processes in an organization by analyzing the existing software artifacts. The objective is to discover the business forces that motivated the construction of the enterprise information systems. On the one hand, traditional archeologists investigate several artifacts and situations, trying to understand what they are looking at, i.e., they must understand the cultural and civilizing forces that produced those artifacts. Similarly, a business process archeologist analyzes different legacy artifacts such as source code, databases and user interfaces and then tries to learn what the organization was thinking to understand why the organization developed the information system in a particular way. Business process archeology initiative is being progressively supported by new reverse engineering techniques and tools to retrieve and elicit the embedded business knowledge. Particularly, this paper focuses on MARBLE™ (*Modernization Approach for Recovering Business processes from LEgacy Systems*) [15]. MARBLE™ is an extensible framework based on the ADM (Architecture-Driven Modernization) initiative proposed by the OMG (Object Management Group). ADM advocates carrying out reengineering processes by considering model-driven development principles.

In addition, MARBLE™ supports the KDM (Knowledge Discovery Metamodel) [10] standard proposed by the ADM initiative. KDM enables the representation and management of knowledge extracted by means of reverse engineering from all the different software artifacts of legacy systems in an integrated way. Thus, that legacy knowledge is gradually transformed into business processes. For this purpose, MARBLE™ is divided into four abstraction levels with three transformations among them (see Figure 1):

Level L0. This level represents the legacy information system in the real world, and is the source system to recover underlying business processes.

Level L1. This level represents several specific models, i.e., one model for each different software artifact involved in the archeology process like source code, database, user interface, and so on. Traditional reverse engineering techniques [5] such as static analysis, dynamic analysis, program slicing, formal concept analysis, and so on, could be used to extract the knowledge from any software artifact and build PSM (Platform-Specific Model) models related to it. These PSM models are represented according to specific metamodels. For example, a Java metamodel may be used to model the legacy source code, or an SQL metamodel to represent the database schema, etc.

Level L2. This level consists of a single PIM (Platform-Independent Model) that represents the integrated view of the set of PSM models at L1. The KDM metamodel is used so that L2 works as a KDM repository that can be progressively populated with knowledge extracted from the different legacy artifacts and information systems of an organization. In addition, L2 is represented in a technological-independent way due to the fact that KDM standard abstract all those details concerning the technological viewpoint (e.g. the program language). The transformation between levels L1 and L2 consists of a set of model transformations implemented using QVT (Query/View/Transformation).

Level L3. Finally, this level depicts, at the end of the archeology process, the business process models retrieved from a legacy system. Business process models at L3 represent a CIM (Computational-Independent Model) and are represented according to the BPMN (Business Process Modeling and Notation). This level closes the conceptual gap between the software architecture views and underlying business rules. The last transformation is based on a set of patterns. When a specific structure is detected in the KDM model at L2, each pattern indicates what elements should be built and how they are interrelated in the business process model at L3 [16]. This pattern matching is implemented through a QVT transformation [17].

The obtained models are a first sketch of the business process, which can be refined by business experts. This is due, for instance, to the fact that not all parts of current business processes are executed by legacy information systems, i.e., there are some manual business activities. Although experts post-intervention can be necessary, the first version of business processes, compared with business process redesign by business experts from scratch, represents a more efficient and less error-prone solution to get business process archeology. In addition, the business process redesign by experts from scratch might discard meaningful business knowledge that is only embedded in legacy information systems.

III. Implementation Details

MARBLE™ technique is supported by a tool with the same name. MARBLE™ tool supports entirely the three model transformations presented between the four MARBLE™ levels. In addition, MARBLE™ allows maintainers and business experts to visualize and modify the first sketch of business processes retrieved after the last transformation.

MARBLE™ tool was initially developed as a desktop application. However, due to the extensible nature of MARBLE™, it is now online available as an *Eclipse*™ plugin [1]. This fact ensures, in the future, easy extension and integration with other maintenance tools.

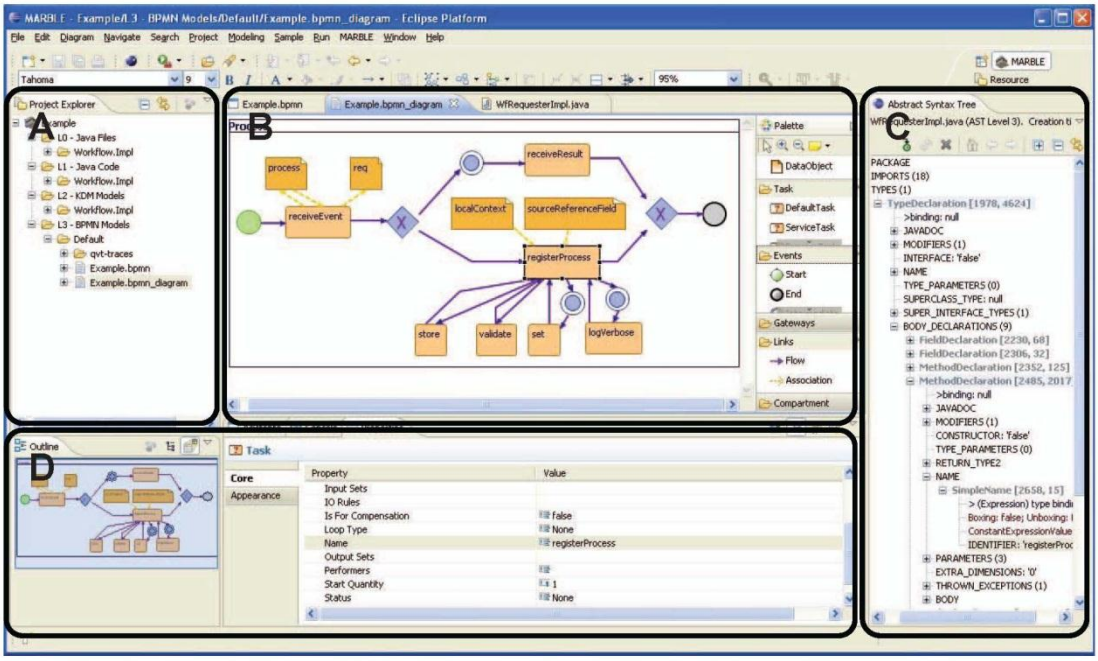


Figure 2. Screenshot of MARBLE™ tool with modules highlighted.

A. Technologies involved

Due to the fact that MARBLE™ technique provides a generic and extensible framework, MARBLE™ tool is firstly developed to support Java-based legacy systems. The tool is implemented through three key technologies. The first technology is *JavaCC*, which is a parser and scanner generator for Java. It is used to develop a Java static analyzer in the first transformation to obtain code models. Parsers for other programming languages may be developed in the future using *JavaCC* technology.

The second technology consists of two related *Eclipse*™ frameworks: EMF (Eclipse Modeling Framework) and GMF (Graphical Modeling Framework). EMF is a modeling framework and code generation facility for building tools and other applications based on ECORE-compliant models. *ECORE* is a meta-metamodel, i.e. it is the metamodel proposed by the *Eclipse*™ platform to define metamodels. EMF provides tools to automatically produce, from an ECORE-compliant metamodel, the source code of an editor to enable viewing and editing of the respective models. Moreover, GMF is used together with EMF to generate graphical editors from *ECORE*-compliant metamodels.

Finally, the third technology is QVT, which is a language specially developed to formalize model transformations. Model transformations involved in MARBLE™ tool are particularly implemented using QVTr (QVT Relations, the declarative language part of the QVT). In addition, model transformations are executed through *MediniQVT*™, an open source QVTr transformation engine [9]. This transformation engine is integrated within the MARBLE™ plugin.

B. Main Functionalities

MARBLE tool functionalities are organized into four main modules. The first module is the project explorer (see Figure 2 A) which groups the models within each MARBLE level. Each model transformation can be done by selecting a whole level or a particular set of models.

The second model is the editor frame (see Figure 2 B) which visualizes the different kinds of models (i.e., java files, code models, KDM models and BPMN models). For example, the screenshot shows a business process model in the graphical editor, which additionally provides a palette of elements to facilitate the graphical edition of the model.

The third module provides a special tree editor with which visualize code models at L1 obtained by the *Java* parser developed (see Figure 2 C). This editor additionally enable feature location since it links code model elements of the abstract syntax tree with the java source code (i.e., an element of the code model can be located by clicking on a certain line of source code).

Finally, the fourth module provides additional relevant information about the particular model opened in the second module (see Figure 2 D). This module particularly provides an outline view as well as a property set view.

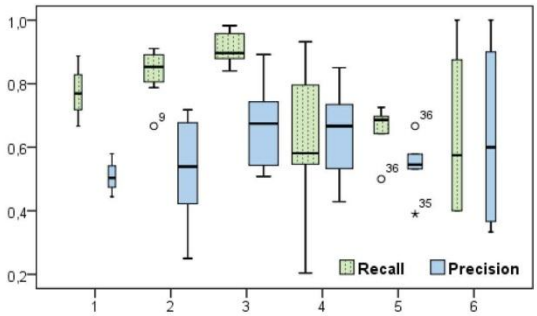


Figure 3. Effectiveness summary of case studies

IV. Case Studies

MARBLE has been applied to several industrial case studies to recover business processes from a wide variety of legacy information systems. The conduction of these industrial case studies has allowed improving the tool and refining the MARBLE technique. So far, MARBLE has been used with six legacy systems in total: (i) a system managing a Spanish author organization; (ii) an open source CRM (Customer Relationship Management) system; (iii) an enterprise information system of the water and waste industry; (iv) an e-government system used in a Spanish local e-administration; (v) a high school LMS (Learning Management System); and finally (vi) an oncological evaluation system used in Austrian hospitals.

These studies evaluate the effectiveness and efficiency of the MARBLE technique applied through the tool. On one hand, effectiveness is measured through precision and recall. Precision measures the exactness or fidelity of the business processes recovered, whereas recall measures their completeness. These measures are computed regarding retrieved tasks and other related business process elements such as sequence flows, data objects and gateways. On the other hand, efficiency is evaluated through the time spent on the recovery as well as the MARBLE scalability to larger legacy information systems. Figure 3 summarizes results obtained from case studies regarding effectiveness. Precision and recall values vary from a system to another, although the value trend is a recall higher than precision. This means that MARBLE retrieves a great number of business activities although a few of them could be erroneous. In addition, a little set of activities could not be retrieved. Anyway, the results are appropriate regarding benchmark values around 0.5 [13].

V. Conclusions

MARBLE is a tool to retrieve business processes from legacy source code. MARBLE provide an extensible framework and it is therefore implemented as an Eclipse™ plugin. So far, MARBLE provides a *Java* parser to obtain code model, which are transformed and integrated in a model repository according to the KDM standard. After that, KDM model are transformed to business process models by applying business pattern recognition. MARBLE has already been applied to real-life modernization projects where business process recovery was a mandatory requirement to preserve the embedded business knowledge.

Acknowledgment

This work was supported by the Spanish FPU Programme, and the R&D projects ALTAMIRA (JCCM, PII2I09-0106-2463), MOTERO (JCCM and FEDER, PEII11-0366-9449) and PEGASO/MAGO (TIN2009-13718-C02-01).

References

[1] Alarcos Research Group. Business Process Archeology. 2011 [cited 2011 10-06-2011]; Available from: http://www.businessprocessarcheology.org/.

[2] Bacchelli, A., et al., Benchmarking Lightweight Techniques to Link E-Mails and Source Code, in WCRE'09. 2009. p. 205-214.

[3] Bernardi, M.L., Reverse engineering of Aspect Oriented Systems to support their comprehension, evolution, testing and assessment, in CSMR'08. 2008, IEEE: Athens, Greece. p. 290-293.

[4] Cai, Z., X. Yang, and W. Wang, Business Process Recovery for System Maintenance - An Empirical Approach, in ICSM'09. 2009, IEEE Computer Society. p. 399-402.

[5] Canfora, G. and M. Di Penta. New Frontiers of Reverse Engineering. in Future of Software Engineering (FOSE'07). 2007: IEEE C. S.

[6] Cleve, A. and J.L. Hainaut. Dynamic Analysis of SQL Statements for Data-Intensive Applications Reverse Engineering. in WCRE'08. 2008.

[7] Di Francescomarino, C., A. Marchetto, and P. Tonella, Reverse Engineering of Business Processes exposed as Web Applications, in CSMR'09. 2009, IEEE C. S. p. 139-148.

[8] Foutse, K., SQUAD: Software Quality Understanding through the Analysis of Design, in WCRE'09. 2009. p. 303-306.

[9] ikv++, Medini QVT. 2008, ikv++ technologies ag.

[10] ISO/IEC, ISO/IEC DIS 19506. Knowledge Discovery Meta-model (KDM), v1.1 (Architecture-Driven Modernization). 2009, ISO/IEC. p. 302.

[11] Khusidman, V. and W. Ulrich, Architecture-Driven Modernization: Transforming the Enterprise. 2007, OMG. p. 7.

[12] Lewis, G.A., D.B. Smith, and K. Kontogiannis, A Research Agenda for Service-Oriented Architecture (SOA): Maintenance and Evolution of Service-Oriented Systems. 2010, Software Engineering Institute. p. 40.

[13] Lucrédio, D., R. de M. Fortes, and J. Whittle, MOOGLE: A Model Search Engine, in Model Driven Engineering Languages and Systems, K. Czarnecki, et al., Editors. 2008, Springer. p. 296-310.

[14] Paradauskas, B. and A. Laurikaitis, Business Knowledge Extraction from Legacy Information Systems. Journal of Information Technology and Control, 2006. 35(3): p. 214-221.

[15] Pérez-Castillo, R., I.G.-R. de Guzmán, and M. Piattini, Business Process Archeology using MARBLE. Information and Software Technology, 2011. 53: p. 1023–1044.

[16] Pérez-Castillo, R., et al., Business Process Patterns for Software Archeology, in 25th Annual ACM Symposium on Applied Computing (SAC'10). 2010, ACM: Sierre, Switzerland. p. 165-166.

[17] Pérez-Castillo, R., I. García-Rodríguez de Guzmán, and M. Piattini, Implementing Business Process Recovery Patterns through QVT Transformations, in International Conference on Model Transformation (ICMT'10). 2010, Springer-Verlag: Málaga, Spain. p. 168-183.

[18] Van der Aalst, W.M.P., et al., ProM : the process mining toolkit, in 7th International Conference on Business Process Management (BPM'09) - Demonstration Track. 2009, Springer-Verlag: Ulm, Germany. p. 1-4.

[19] Zou, Y. and M. Hung, An Approach for Extracting Workflows from E-Commerce Applications, in 14th International Conference on Program Comprehension. 2006, IEEE Computer Society. p. 127-136.

CHAPTER 5

Empirical Validation

> *This technique has been validated over a two-year period in several industrial modernization projects.*

5. EMPIRICAL VALIDATION

This chapter focuses on the empirical validation of the research proposal presented in Chapter 3. Although most of the papers presented in that chapter contain industrial case studies, this chapter addresses the whole empirical assessment by aggregating individual results. The aggregation and analysis of individual case studies is carried out by means of both the Bender method and meta-analysis. Both research methods are mentioned in Section 1.3.

5.1. Precision and Recall Indicators

This section applies the Bender method in order to obtain indicators, i.e., reference values, with which to measure the precision and recall used in all the case studies conducted. While precision is used to evaluate the exactness of the business process models discovered, recall is related to the completeness of the models obtained. Both measures are evaluated between 0 and 1. Indicators are necessary to compare results obtained from different case studies and different techniques, since there is absence of thresholds to check whether or not the precision and recall values obtained in a particular study are appropriate.

This chapter consists of a paper published in the proceedings of the 5th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. This conference is considered as CORE A (the top quality level) according to the ERA Conference Ranking. The reference of this paper is *Pérez-Castillo, R., L. Sánchez-González, F. García, M. Piattini, and I. García Rodríguez de Guzmán, Obtaining Thresholds for the Effectiveness of Business Process Mining, in 5th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'11). 2011, IEEE Computer Society: Banff, Alberta, Canada. p. 453-462.*

2011 International Symposium on Empirical Software Engineering and Measurement

Obtaining Thresholds for the Effectiveness of Business Process Mining

[1] Ricardo Pérez-Castillo, [1] Laura Sánchez-González, [2] Mario Piattini, [1] Félix García and [1] Ignacio García-Rodriguez de Guzmán
[1] University of Castilla-La Mancha
Paseo de la Universidad 4 13071, Ciudad Real, Spain
{ricardo.pdelcastillo, laura.sanchez, felix.garcia, ignacio.grodriguez}@uclm.es
[2] Alarcos Quality Center
Camino de Moledores s/n, 13051 Ciudad Real, Spain
mario.piattini@alarcosqualitycenter.com

Abstract
Business process mining is a powerful tool to retrieve the valuable business knowledge embedded in existing information systems. The effectiveness of this kind of proposal is usually evaluated using recall and precision, which respectively measure the completeness and exactness of the retrieved business processes. Since the effectiveness assessment of business process mining is a difficult and error-prone activity, the main hypothesis of this work studies the possibility of obtaining thresholds to determine when recall and precision values are appropriate. The business process mining technique under study is MARBLE, a model-driven framework to retrieve business processes from existing information systems. The Bender method was applied to obtain the thresholds of the recall and precision measures. The experimental data used as input were obtained from a set of 44 business processes retrieved with MARBLE through a family of case studies carried out over the last two years. The study provides thresholds for recall and precision measures, which facilitates the interpretation of their values by means of five linguistic labels that range from low to very high. As a result, recall must be high (with at least a medium precision above 0.56), and precision must also be high (with at least a low recall of 0.70) to ensure that business processes were recovered (by using MARBLE) with an effectiveness value above 0.65. The thresholds allowed us to ascertain with more confidence whether MARBLE can effectively mine business processes from existing information systems. In addition, the provided results can be used as reference values to compare MARBLE with other similar business process mining techniques.

Keywords: **Business Process, Process Mining, Thresholds, Bender Method, Case Study.**

I. INTRODUCTION

Business processes are an essential asset for organizations. They provide a means for mapping business objectives regarding how to carry out better operations, generating added value for customers [43]. Thereby, business processes improve the competitiveness level of the organizations that consider and manage them [8].

At this time, most business processes are automated by enterprise information systems [18]. Thus, business processes must sometimes be recovered from existing information systems for two main reasons. First, some organizations do not explicitly manage their business processes, and their existing information systems embed valuable business knowledge that is not present anywhere else [19]. Second, existing information systems evolve faster than business processes owing to uncontrolled software maintenance overtime, and obsolete and misaligned business processes raise an important concern for organizations [16].

The solution to retrieve business processes from existing information systems is known as 'business process mining' [41], and it is used by organizations to eradicate, or at least mitigate, these problems.

Many business process mining approaches and techniques exist in the literature [1]. Some business process mining proposals have been empirically validated in terms of their effectiveness, i.e., the degree of quality of the obtained business processes. So far, several measures have been used to measure the effectiveness of process mining techniques by comparing the discovered business processes with respect to the reference processes (i.e., business processes constructed in the design stage). Of the existing effectiveness measures, the most notable are *recall* and *precision*, two measures adapted from the research area of information retrieval, which have been used in a great number of empirical studies and experiments. Intuitively, precision indicates how exact a mined business process is, while recall indicates how complete a process is.

Despite the fact that recall and precision have been used in several empirical studies, there is no reference threshold to indicate which precision and recall values are appropriate for business process mining scenarios. For instance, some questions can arise such as *"How good is 0.67 precision?"* or *"Is that value an acceptable result?"* and these currently cannot be answered. This problem implies that the precision and recall values obtained in empirical studies cannot therefore be conveniently interpreted and this limits the usefulness of the information provided to assess the effectiveness of a particular business process mining technique.

This paper examines this challenge and obtains benchmark thresholds to interpret the precision and recall measure values. The thresholds were obtained by applying the Bender method [3] to the data extracted from a family of case studies in which MARBLE (*Modernization Approach for Recovering Business processes from LEgacy systems*), a business process mining technique proposed in previous works [29, 32], was validated. The family of case studies was carried out over the last two years and is composed of six industrial case studies involving six real-life information

978-0-7695-4604-9/11 $26.00 © 2011 IEEE
DOI 10.1109/ESEM.2011.64

IEEE computer society

systems. As a result, the precision and recall measures were turned into indicators, by associating *decision criteria* to them [14]. In order to obtain the thresholds, the Bender method was applied to the data about MARBLE effectiveness extracted from the case studies.

The remainder of this paper is organized as follows. Section II provides the background about precision and recall measures. Section III presents related work. Section IV explains in detail the data obtained from the family of case studies which were used to obtain thresholds. Section V describes how the *Bender* method was applied to obtain the thresholds and the obtained results are presented. These thresholds are applied to interpret the data from the case studies, as illustrated in Section VI. Finally, Section VII discusses the main conclusions of this study as well as future work.

II. Information Retrieval Measures

Recall and precision are two measures raised from the information retrieval research field, and they have been used for many years [42]. In addition, both recall and precision measures have usually been used to evaluate the effectiveness of business process mining techniques or algorithms as well as other kinds of reverse engineering methods [37]. As a result, these measures not only evaluate the obtained business processes, but they evaluate the effectiveness of the method to recover business processes.

The evaluation of recall and precision consists of measuring the difference between the obtained and desired results. These measures provide a number between 0 and 1 without unity. Recall and precision measure the similarity between a mined business process *M* and a reference business process *R*. Recall indicates what proportion of *R* is present in *M* (i.e., how complete *M* is), while precision indicates what proportion of *M* matches *R* (i.e., how exact *M* is). On the one hand, recall is the number of relevant elements mined over the total number of existing relevant elements in a business process. On the other hand, precision is scored as the number of relevant elements mined over the total number of elements of a certain mined business process.

A particular element is considered *relevant* if that element faithfully represents a piece of business operation or business behavior of the organization in the real world. To evaluate these measures, various business process elements (e.g. activities, sequence flows, data objects, etc.) were used as the unit to analyse the relevance. Despite this variety, the business activity (or business task) is the most commonly used element.

As a consequence, the recall measure (1) is defined as the number of retrieved relevant tasks divided by the total number of relevant tasks (i.e., divided by the retrieved and non-retrieved relevant tasks). Moreover, a precision measure (2) is defined as the number of retrieved relevant tasks divided by the total number of retrieved (relevant and non-relevant) tasks.

$$RECALL = \frac{\{retrieved\ relevant\ tasks\}}{\{retrieved\ relevant\ tasks\} + \{non\ retrieved\ relevant\ tasks\}} \quad (1)$$

$$PRECISION = \frac{\{retrieved\ relevant\ tasks\}}{\{retrieved\ relevant\ tasks\} + \{retrieved\ non\ relevant\ tasks\}} \quad (2)$$

$$Fmeasure = \frac{(1+\alpha) \cdot Recall \cdot Precision}{\alpha \cdot \text{Precision} + \text{Recall}} \quad (3)$$

Although recall and precision are appropriate, there is an inverse relationship between both measures. As a result, these measures are rarely used in an isolated way, and recall and precision are therefore usually combined into a single measure known as the F-measure. The F-measure (3) consists of a weighted harmonic mean of recall and precision. The F-measure makes it possible to modify the importance of precision and recall by means of the value. The value is usually 1, which means that precision and recall have the same weight in the F-measure formula. This measure is interesting, because it gives a low result for business process mining techniques, which improves precision or recall exclusively to the detriment of the other [25].

All business process mining proposals usually try to maximize the F-measure by maximising both recall and precision measures, which is the desirable result. Indeed, the precision value should, ideally, always be 1 for any recall value but, according to [12], this is not possible in practice due to their inverse relationship (see Figure 1). Because of this, business process mining techniques obtain quite different results, for instance a high precision and a low recall or vice versa. For this reason, reference thresholds are needed to know when the effectiveness of a particular business process mining technique is actually good.

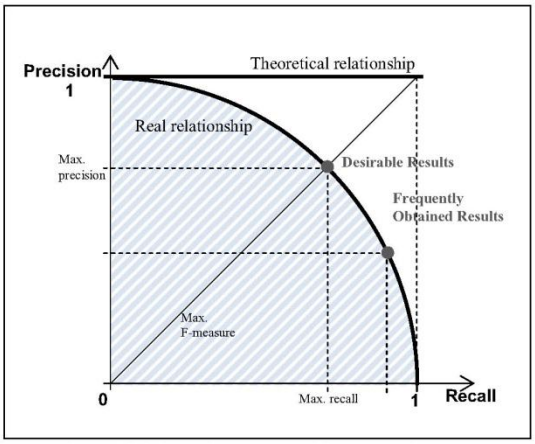


Figure 1. Relationship between precision and recall measures

III. Related Work

This section first presents some existing techniques for business process mining which use recall and precision measures to evaluate their effectiveness. Then, the section summarizes some threshold definition proposals.

A. Business Process Miners Precision and Recall

There are many proposals for business process miners in the literature which use precision and recall measures to evaluate their effectiveness. *Lo and Khoo* [36] propose a QUARK (QUality Assurance framewoRK), which enables assessments of the effectiveness of specification miners based on finite state machines. This framework uses the k-tails algorithm [5], which considers all the sub-sequences of k length from each node to score precision and recall measures. This framework has reported an average precision of 0.10 and an average recall of 0.90. The disadvantage of this framework is that it uses the finite state machine format to represent the mined processes, instead of Business Process Modeling and Notation (BPMN) [28] or another standardized format.

Pérez-Castillo et al. [33] present a case study to compare the effectiveness of two related business process mining techniques. The first technique statically analyses the source code, and the second one analyses traces obtained during system execution. The study reports that the second technique has higher precision and recall values, which are respectively 0.70 and 0.77.

Lou et al. [20] propose a business process mining algorithm using event traces recorded during system execution. This work provides a set of experiments that reports a precision value around 0.50 and an average recall of 0.99. Following a similar approach, *Povzner et al.* [35] propose Autograph, a framework to extract data file signatures in workflows. This approach examines traces of file accesses, finds repeated and correlated accesses, and infers which files most likely belong to the same workflow. *Povzner* and colleagues carried out an experiment using benchmark programs and reached precision and recall values of 0.65 and 0.85 respectively.

Some approaches focus on recovering patterns concerning the source code architecture. For instance, *Asencio et al.* [2] provide a program analysis tool to automatically recognise the use of software design patterns found in object-oriented code. This framework is applied to an ACE system (Adaptive Communications Environment). The precision and recall values obtained in this evaluation are 0.68 and 0.71 respectively. Another study that follows the reverse engineering of design patterns from source code is proposed by *Philippow et al.* [34]. They provide a survey of pattern search methods, which have been applied to several systems. This work reports precision and recall values of 0.37 and 1.0 in some cases. *Yeh et al.* [45]carried out an empirical study of a miner, focusing on the recovery of object relationships from source code. This study reports an average precision of 0.74 and a recall of 0.81.

Other works address the search in business process model repositories, instead of recovery business process models from information systems. For instance, *Lucrédio et al.* [21] provide MOOGLE, a model search engine to recover several kinds of models (not only business process models) from large model repositories. This work presents the result of an experiment using MOOGLE, which provides precision and recall values of 0.24 and 0.53. In addition, *Lucrédio et al.* [21] take into account similar empirical studies such as [13, 15, 44] to fix a benchmark value of 0.50 for both recall and precision measures.

Despite the aforementioned proposals can consider different source datasets to asses precision and recall, TABLE I compares the precision and recall obtained in each work. Recall is higher than precision in all of the proposals. This trend is due to the fact that any reverse engineering method, such as business process mining, makes it possible to recover a significant amount of information. However, this does not mean that the appropriate information is automatically recovered. Indeed, some information may not be available to be retrieved by means of the business process mining technique. For this reason, obtaining higher precision is a key challenge.

TABLE I. PRECISION AND RECALL OBTAINED IN THE LITERATURE

	Precision	Recall
Lo and Khoo [36]	0.10	0.90
Pérez-Castillo et al.[33]	0.70	0.77
Lou et al.[20]	0.50	0.99
Povzner et al.[35]	0.65	0.85
Asencio et al.[2]	0.68	0.71
Philippow et al.[34]	0.37	1.00
Yeh et al.[45]	0.74	0.81
Lucrédio et al.[21]	0.24	0.53

These proposals focus on business process mining techniques or algorithms that use precision and recall measures to evaluate their effectiveness. However, a wide variety of measures are used to measure different features of business processes [39]. For instance, the control-flow complexity measure/(CFC) [7] is used to calculate the degree of complexity of process models from a decision-node perspective. The main idea is that the high complexity of process models produces difficulties in terms of understanding the model, because of an increase in errors, deadlocks, bottlenecks, etc. The use of a CFC measure allows designers to improve process models, reducing the time needed to read and understand processes in order to adapt them to new requirements. Other interesting measures include error probability measures [23]. This group of measures calculates the likelihood of finding errors in business process models, specifically 13 measures related to process model elements: number of nodes, diameter, density, depth, sequentiality, separability, cyclicity, average and max degree of connectors, mismatch connector, connector heterogeneity, connectivity coefficient and concurrency. However, this kind of measure focuses on mined business processes, but does not allow the effectiveness of the mining method to be measured.

B. Threshold Definition Proposals

It is possible to find several proposals about the definition of thresholds in the literature, especially in the medical field. Unfortunately, for calculation of threshold values no standard method is available. Some authors base

threshold values on experience, which is the case with *McCabe* [22], *Nejmeh* [26] and *Coleman et al.* [10]. Others adapt some medical methods for software engineering, for example *Shatnawi* [40], who uses the Bender method [3] in order to obtain thresholds for object-oriented measures. In that work, the author defines thresholds to study the relationship between some object-oriented measures (e.g., Coupling Between Objects (CBO), Response For Class (RFC), Weighted Methods Complexity (WMC), Depth of Inheritance Hierarchy (DIT)) as well as error-severity categories, concluding by demonstrating the utility of the Bender method for this purpose. On the other hand, *Berlarbi et al.* [4] extract thresholds in order to predict which classes are likely to contain faults and test threshold effects in subsets of the *Chidamber and Kemerer* measures [11]. However, these results are only valid for the measures used by the authors, since other models may potentially produce different results.

IV. Data Collection

This section presents the data we used in this study to obtain the benchmark thresholds by applying the Bender method. The case studies were carried out over the last two years with six different information systems to validate the effectiveness of MARBLE. To improve the rigor and validity of the case studies, the guidelines proposed by *Runeson* and *Höst* [38] and *Brereton el al.* [6] were followed. The following sub-sections summarise how the stages of the protocol were followed.

A. The Object of Study

The *object of study* is MARBLE [29], which is a model-driven framework to recover business process models from existing information systems. MARBLE considers different models at different abstraction levels and a set of model transformations between them. A supporting tool is additionally provided in order to automate the technique and facilitate its adoption.

Specifically, MARBLE defines four kinds of models at four different abstraction levels. *L0* is the lowest level of abstraction since it represents the existing information system in the real world as a set of different software artifacts (e.g. source code, database, documentation, etc). *L1* contains different platform-specific models (PSM) depicting the different software artifacts of the LIS. *L2* integrates all the specific *L1* models into a platform-independent model (PIM), which is represented according to the Knowledge Discovery Metamodel (KDM) [17]. KDM enables the representation and management of the knowledge extracted by means of reverse engineering from all the different software artifacts of legacy information systems. Finally, *L3* depicts the discovered business processes, which are represented according to the BPMN metamodel [28].

Moreover, MARBLE defines three model transformations between the four levels (see Figure 2). First, the L0-to-L1 transformation obtains PSM models from each legacy software artifact using a specific metamodel for each artifact. Traditional reverse engineering techniques such as static analysis or dynamic analysis are used to extract the needed knowledge. Second, the L1-to-L2 transformation (see Figure 2) consists of a set of model transformations (e.g. implemented using the QVT (Query/View/Transformation) language [27]) to obtain a KDM model built from the PSM models at L1. Finally, the L2-to-L3 transformation obtains the current business process model. This transformation is based on a set of business patterns [30], which define the transformation rules between levels L2 and L3. In addition, this last transformation can be implemented through QVT rules [31].

MARBLE assesses precision, recall and F-measure by considering task as relevant when they are retrieved in the same order, with the same related sequence flows and data objects [29]. This evaluation is carried out by business experts once business processes were retrieved.

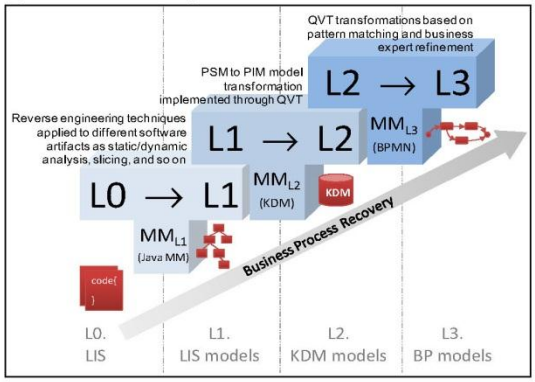


Figure 2. MARBLE Overview

B. Design

While the *object of study* is MARBLE, the *purpose of this study* is the evaluation of its effectiveness. For this purpose, the family of case studies defines the following research question: *"Can MARBLE effectively recover business processes from existing information systems?"*

Each case study in the family follows the *embedded case study* design. This means that each case study consists of a single case, (i.e. it focuses on a single information system), but in addition, the study considers several analysis units within the case. The analysis units are the different business processes retrieved from each information system, which represent the independent variable in each study.

Each study consists of the analysis of all the retrieved business processes in order to answer the research question. Therefore, the family of six case studies uses recall, precision and F-measure (c.f. Section II) to provide quantitative answers to the proposed research question.

C. Case Selection

Cases under study cannot be randomly selected. Thus, the case selection protocol defines a list of four criteria to choose suitable cases to be studied. The criterion *C1* guarantees that the existing information system selected is an enterprise system that supports the business operation of an organisation. *C2* ensures that the selected system is truly a legacy information system. This criterion considers the number of system modifications that altered the business

processes. *C3* ensures that the system is not a *toy program*, since it defines a threshold of 20,000 lines of source code. Finally, *C4* guarantees that the system is based on the Java platform, since the supporting tool was developed for Java-based systems. After evaluating, according to these criteria, six information systems were selected from eleven available systems, given by partner companies. TABLE II shows the name of each system, a brief description, the type of architecture, and the size in thousands of lines of source code.

TABLE II. LEGACY INFORMATION SYSTEMS UNDER STUDY

Id	Name	Description	Architecture	Size (KLOC)
S1	AELG-Members	Supports the administration of an organization of authors	Desktop application	23.5
S2	Tap CRM	A sales force automation tool for sales management	Web application	49.6
S3	VillasanteLab	Manages a laboratory in the water and waste industry	Web application	28.8
S4	XuntaEadmin	Supports the electronic administration of a Spanish regional ministry	E-government system	320.2
S5	SIXA	A learning management system	Web application	140.6
S6	CHES	A computer-based health evaluation system for the oncology area	Desktop application	619.8

D. Collected Data

TABLE III provides the data collected during the execution of the case studies. TABLE III shows: (i) the business process identifier; (ii) the case study identifier; (iii) the retrieved business process name; (iv) the precision (v) recall, and (vi) F-measure values for each retrieved business process.

These data constitute the input necessary for the application of the Bender method in order to obtain the goodness thresholds for both recall and precision measures (see Section V).

E. Analysis and Interpretation

The collected data were analysed to obtain evidence chains from data to answer the research question (c.f. sub-section B). In order to answer the proposed question, Figure 3 shows the box chart for recall and precision measures. The means of the distributions of the precision measure (0.51, 0.53, 0.66, 0.64, 0.54 and 0.63) are always lower than the recall means (0.77, 0.83, 0.91, 0.65, 0.66 and 0.64). Higher recall values mean that the proposed technique retrieves very complete processes (i.e., it retrieves most of the tasks from the real-world business processes). Nevertheless, the lower precision value means that the business process mining technique is imprecise (i.e., the number of retrieved non-relevant tasks is very high) (see Figure 3). These recall and precision values respectively provide F-measure means for each case study of 0.61, 0.64, 0.76, 0.63, 0.59 and 0.63.

TABLE III. RESULTS OBTAINED FOR THE FAMILY OF CASE STUDIES

Subject ID	Study ID	Business Process Model Name	Recall	Precision	F-Measure
1	1	Categories Management	0.769	0.580	0.661
2	1	Author Management	0.888	0.504	0.643
3	1	Reporting	0.667	0.444	0.533
4	2	Security Management	0.882	0.435	0.583
5	2	Administration	0.788	0.578	0.667
6	2	Chemical Analysis Management	0.911	0.638	0.750
7	2	Chemical Calibration Management	0.824	0.718	0.767
8	2	User Management	0.842	0.250	0.386
9	2	Chemical Dilution Management	0.667	0.410	0.508
10	2	Reporting	0.864	0.717	0.784
11	2	District Management	0.900	0.500	0.643
12	3	Administration	0.873	0.515	0.648
13	3	Social Protection Floor Mgmt.	0.914	0.660	0.766
14	3	Document Management	0.885	0.697	0.780
15	3	Rental House Management	0.961	0.762	0.850
16	3	Renovation Document Registration	0.983	0.828	0.899
17	3	House Applicant Management	0.955	0.892	0.922
18	3	Personal File Management	0.892	0.552	0.682
19	3	Rural House Management	0.840	0.534	0.653
20	3	Developer Management	0.853	0.596	0.702
21	3	Renovation Management	0.968	0.689	0.805
22	3	Emancipation Grant Management	0.899	0.508	0.649
23	3	Second-Hand House Management	0.894	0.724	0.800
24	4	Schedule Management	0.820	0.702	0.756
25	4	Qualification Management	0.918	0.767	0.836
26	4	Class Group Management	0.564	0.667	0.611
27	4	Evaluation	0.204	0.636	0.309
28	4	Remark Management	0.771	0.851	0.809
29	4	Reporting	0.731	0.667	0.698
30	4	Notice Management	0.581	0.556	0.568
31	4	Permission Management	0.571	0.511	0.539
32	4	Student Management	0.529	0.486	0.507
33	4	Subject Management	0.529	0.429	0.474
34	4	Teacher Management	0.932	0.786	0.853
35	5	Provider Management	0.697	0.390	0.500
36	5	Product Management	0.500	0.667	0.571
37	5	Event Management	0.676	0.532	0.595
38	5	Sending Management	0.725	0.578	0.643
39	5	Reporting	0.643	0.545	0.590
40	5	Relationship Management	0.698	0.545	0.612
41	6	Patient Admission	0,400	0,400	0,400
42	6	Data Collection Management	1.000	0.400	0.571
43	6	Data Analysis Management	0.800	1.000	0.889
44	6	Patient Stay Management	0.333	0.750	0.462
		Mean	0,763	0,604	0,656
		Standard Deviation	0,180	0,156	0,145
		Median	0.579	0.822	0.649

The obtained results are aligned with other business process mining proposals (cf. Section III), which achieve recall values higher than precision values. The lower precision is due to two main reasons.

First, all reverse engineering techniques, such as business process mining techniques, lose specific knowledge when the information is represented in a higher abstraction level. For the particular business process mining technique, the low precision is a consequence of the significant number of non-relevant tasks. In turn, this is due to the fact that some tasks

are basically obtained from the source code and are related to the technical nature, and do not therefore represent any piece of the embedded business knowledge. Second, an inverse relationship exists between both recall and precision measures. Taking into account this circumstance, automated business process mining techniques achieve high recall more easily than high precision.

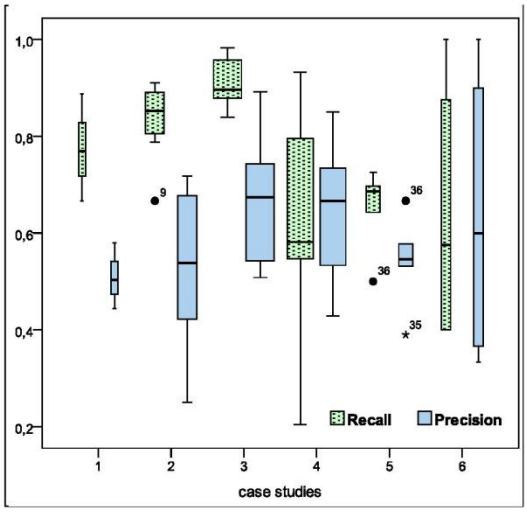


Figure 3. Box chart of recall and precision for each case study

Regardless, the aforementioned analysis of the obtained data was not sufficient to answer the research question. If we consider the reference value of 0.5 as used by *Lucrédio et al.* [21], the research question could be positively answered, since recall and precision are above 0.5. However, that reference value was selected heuristically, which limits the reliability of the conclusions. For this reason, in the present work, goodness thresholds for recall and precision were achieved by applying the Bender method.

V. Threshold Definition

A. The Bender Method

In this work, we used the Bender method to extract thresholds for both recall and precision measures. Thresholds help us to assess the numerical results of these measures, creating the probability of finding adequate results. This method was developed to discover thresholds in epidemiological studies. The Bender method assumes that the risk of an event happening is constant below a specific value (i.e., the threshold) and increases according to a logistic equation. The Bender method defines a "Value of an Acceptable Risk Level" (VARL), defined as (4).

$$VARL = \frac{1}{\beta} \cdot \left(ln\left(\frac{p_0}{1 - p_0} - \alpha \right) \right) \quad (4)$$

In equation (4), p_0 is the probability of an event occurring. This value is indicated by the author and can vary from 0 to 1. For example, $p_0 = 0.6$ indicates that there is a probability of 0.6 for considering recall or precision as appropriate. On the other hand, α and β are coefficients of a logistic regression equation, shaped as (5).

$$y = \alpha + \beta \cdot measureX \quad (5)$$

The independent variable in the logistic regression equation is the measure or measures (denominated as *measureX* in (5)), which we want to extract their thresholds. The dependent variable must be a binary variable and is usually called *goodness*. This dependent variable indicates if the independent variables are considered good, taking into account some factors such as error degree, suitability, and so on.

Sample size in regression analysis is extremely important and it can determinate if the study has a significant effect. An approximation of sample size can be calculated though four parameters as indicated in [9]: alpha probability level, number of predictors in the linear model, anticipated effect size and desired statistical power level. An adequate sample size ensures more reliable results.

The Bender method works by extracting different VARLs with the specific values of $p_{0,2}$, $p_{0,4}$, $p_{0,6}$, $p_{0,8}$. These four values of probability divide the measure distributions into five groups, which have associated linguistic labels for an accurate interpretation, ranging from "very low" to "very high". The application of the Bender method in this work is described in the next section.

B. Application of the Bender Method to Obtain Precision and Recall Thresholds

First, to apply the Bender method, the dependent and independent variables must be selected. The independent variables are the different measures analysed in this work, i.e., recall and precision. The dependent variable is a binary variable created specifically for this purpose. This variable is called 'goodness' and indicates whether both the recall and the precision are considered good, taking in account the F-measure value. As described in Section II, the F-measure evaluates the relationship between recall and precision, because the idea is to obtain their maximum value at the same time. For this reason, we considered that in previously analysed data (c.f. Section IV), the retrieved business process which obtains an F-measure higher than the median of its own distribution is considered valid. Thereby, we consider a binary variable denominated as *goodness* to be 1 and 0 when the F-measure value is respectively under or above the median. For example, for a specific value of the F-measure of 0.75 (see subject 6 in TABLE III), the recall is 0.91 and

the precision is 0.63. We considered these values as *appropriate* because the F-measure is higher than the median (0.65). In that particular case, the value of the goodness measure is 1. On the other hand, the F-measure is 0.38 when the recall is 0.84 and the precision 0.25 (see subject 8 in TABLE III). Although recall has a high value (which can indicate whether the retrieval is good), the precision value is too low to consider the entire retrieval process as *appropriate*. As a consequence, the *goodness* value is 0 (i.e., not appropriate).

Although the new measure goodness involves an assessment of the measures (recall and precision) the statistical method selected (Bender method) is considered more powerful by the way it not only indicates limit values. The selected method also investigates a threshold effect by the association of probabilities of considering the retrieval process acceptable or not.

A logistic regression analysis was applied to the experimental data provided in TABLE III. The minimum sample size for the experimental data has been calculated with the following configuration according to [9]:

- Alpha probability level, also known as the p-value, is 0.05 to ensure statistical significance.
- Number of predictors in the linear model. In this case, a regression equation for each variable is needed, so the number of predictors is 1.
- The anticipated effect size (f^2) is 0.2 which is a medium value, since by convention effect sizes of 0.02, 0.15 and 0.35 are considered small, medium and large.
- The value of desired statistical power level is 0.8, since by convention this value is always greater than or equal to 0.8.

As a result, a sample size greater than or equal to 41 is adequate in order to obtain a meaningful regression analysis. In our case, 44 data entries are available (see TABLE III).

After applying a logistic regression analysis to the experimental data, we obtained α and β coefficients, which can be used to obtain VARLs values (see TABLE IV).

TABLE IV. α AND β COEFFICIENTS

Results	Precision	Recall
α	-6.77	-12.46
β	11.28	15.73
VARL$p_{0,2}$	0.47	0.70
VARL$p_{0,4}$	0.56	0.76
VARL$p_{0,6}$	0.63	0.81
VARLp_{08}	0.72	0.88

TABLE IV depicts the threshold values, which can be interpreted as follows: "There is a probability of 0.2 for considering the retrieval of a business process acceptable if the measure precision is 0.47 and recall is 0.70".

We assigned linguistic labels to these different groups in order to provide an interpretation for the numerical results, with the following results:

For recall:

- if recall • 0.70 →*very low recall*
- if 0.70 < recall • 0.76 → *low recall*
- if 0.76 < recall • 0.81 →*medium recall*
- if 0.81 < recall • 0.88 →*high recall*
- if recall > 0.88 →*very high recall*

For precision:

- if precision • 0.47 →*very low precision*
- if 0.47 < precision • 0.56 →*low precision*
- if 0.56 < precision • 0.63 →*medium precision*
- if 0.63 < precision • 0.72 →*high precision*
- if precision> 0.72 →*very high precision*

As noted in the earlier sections, precision and recall should be evaluated in conjunction, represented as the F-measure. The idea is to maximize both measures, because when good retrieval occurs, precision and recall have high values. However, case studies reveal that maximising both values involves high costs, and it is difficult to obtain high values for both measures. This leads to combining the precision and recall results to obtain a general evaluation of the retrieval process. TABLE V provides the range of F-measure values for each pair of combined labels of both recall and precision measures.

In TABLE V, it is possible to observe the evaluation of the F-measure through the combination of minimum and maximum precision and recall values for each threshold. Each cell represents the minimum and maximum value of the F-measure that can be achieved with the respective levels of recall and precision. The best choice is when a high F-measure is achieved. TABLE V shows some highlighted cells that respectively indicate the recall and precision thresholds that provide F-measures above 0.65, 0.70 and 0.75. We consider these combinations as the appropriate values for the effectiveness of MARBLE. Thereby, the effectiveness of MARBLE is considered as *acceptable* in two minimal cases: (i) if precision is at least medium when recall is high; or (ii) if recall is at least low when precision is high. As a result, for MARBLE to be considered an effective mining technique, there must be at least (i) a precision above 0.56 in combination with a recall above 0.81; or (ii) a recall above 0.70 in combination with a precision above 0.63. Despite these acceptable values, the most appropriate effectiveness would be achieved when the recall is at least high and precision is very high (see highlighted cells in TABLE V). This means that the best values for recall and precision must be above 0.81 and 0.72, respectively.

TABLE V. ASSESSMENT OF F-MEASURE

		Recall				
		Very low	Low	Medium	High	Very high
Precision	Very low	(0, 0.56)	(0, 0.58)	(0, 0.59)	(0, 0.61)	(0, 0.64)
	Low	(0, 0.62)	(0.56, 0.64)	(0.58, 0.66)	(0.59, 0.68)	(0.61, 0.72)

	Medium	(0, 0.66)	0.62, 0.69)	(0.64, 0.71)	(0.66, 0.73)	(0.68, 0.77)
	High	(0, 0.71)	(0.66, 0.74)	(0.69, 0.76)	(0.71, 0.79)	(0.73, 0.84)
	Very high	(0, 0.82)	(0.71, 0.84)	(0.74, 0.90)	(0.76, 0.94)	(0.79, 1)

F-measure Intervals	≥0.65	≥0.70	≥0.75

VI. Discussion of the Results and Implications

This section discusses the thresholds obtained by applying the Bender method. First, the thresholds for recall and precision are used to evaluate the input data from the family of case studies in Section IV. Finally, the threats to the validity of this study are presented.

A. Interpretation of Experimental Data

The thresholds obtained for the precision and recall measures were used to provide a more confident answer to the research question put forth in Section IV.

If we compare the total mean for precision and recall (see TABLE III) with the thresholds obtained, we observe that both recall and precision measures are *medium*, since the mean of the recall is 0.763 (between 0.76 and 0.81) and the mean of the precision is 0.604 (between 0.56 and 0.63). As a result, the answer to the research question is more finely tuned with this additional information. Now, it is possible to state that the proposed technique retrieves business processes with a medium effectiveness, which means that it does not have the most appropriate performance.

In order to evaluate which case studies reported an appropriate effectiveness, the obtained thresholds can also be applied to the mean of the recall and precision of each case study. TABLE VI shows the interpretation of the results obtained in each case study using the proposed thresholds and linguistic labels. The best results are obtained for the third case study, and the results obtained in the first and second case studies can be considered acceptable. However, the fourth and sixth case studies have high precision, but very low recall. Finally, the effectiveness results obtained in the fifth case study cannot be considered appropriate, since the precision is low and the recall very low.

TABLE VI. Application of thresholds for each case study

		Recall				
		Very low	Low	Medium	High	Very high
Precision	Very low					
	Low	S5		S1	S2	
	Medium					
	High	S4, S6				S3
	Very high					

Additionally, the obtained thresholds may be individually applied to each business process to discover which processes were retrieved with appropriate levels of recall and precision. For instance, when business processes with a low precision and low recall are obtained, they are not considered relevant. These results also can aid in the improvement of business process mining techniques by observing the features common to the inappropriate business processes, leading to certain modifications.

B. Threats to the Validity

This section shows the threats to the validity of this study as well as the list of possible actions to mitigate them. We mainly consider two types of validity: internal and external.

At least two key threats to the *internal validity* exist. The first threat is related to the dependent variable selected to adapt the Bender method. The logical *goodness* variable is selected taking into account the median of the F-measure distribution. In this respect, the concept of *goodness* could be completed by other general characteristics about the retrieval process, for example, the subjective opinion of business experts or their satisfaction with the result, or other internal characteristics of the process such as the structural complexity. However, it is not easy to carry out large empirical studies involving appropriate business experts in each case. In addition, the business expert viewpoint could introduce a bias to those studies.

The second threat to the internal validity is related to the linguistic labels. These labels were established considering five equidistant probabilistic ranges, i.e., 0 to 0.2; 0.2 to 0.4; 0.4 to 0.6; 0.6 to 0.8 and 0.8 to 1.0, by configuring p_0 in the equation (4). However, these ranges could be established considering a different number of ranges as well as considering different sizes for each range of probability. For instance, *Miller* [24] states that seven, plus or minus two, is the most appropriate number to establish different linguistic labels. To mitigate this threat, the study could be replicated by adapting the Bender method using different ranges of probability taking into account expert's opinion according to qualitative research. The results could be then compared to provide strengthened conclusions.

Regarding *external validity*, this study considers the domain of business process mining techniques that use recall and precision to evaluate their effectiveness to be the whole population. Thus, the obtained thresholds could be generalized to this population. However, the experimental data are extracted from the application of a specific technique, MARBLE. This limits the extent to which the thresholds obtained in this study can be generalised to other business process mining techniques based on approaches or principles different to MARBLE. In order to mitigate this threat, the study should be replicated by extending the experimental data population through the consideration of different business process mining techniques. The threshold values extracted through the Bender method are dependent on the experimental data since it uses a logistic regression. For this reason, a larger set of case studies could improve the likelihood of obtaining extrapolated results.

VII. Conclusions and Future Work

A great variety of business process mining techniques and algorithms have been proposed in the literature. The effectiveness of some of these proposals has usually been evaluated using recall and precision, but due to the lack of thresholds for measures, the analysis and interpretation of the obtained results is a difficult and error-prone activity. This paper proposes goodness thresholds for recall and precision values to evaluate the effectiveness of the business process mining technique MARBLE. As a result, five linguistic labels for each measure (very low, low, medium, high, very high) were provided to indicate the goodness of a recall or precision value obtained for a particular case. The results indicate that MARBLE has an appropriate effectiveness when, at least, precision is medium and recall is high or recall is low and precision is high.

Obtaining thresholds for such measures is partially context-specific. However, this work provides a novel approximation which can be applied to other business process mining techniques to obtain their own reference values which can be used to compare their effectiveness. This is an important aspect which can be used to facilitate the selection of the most effective business process mining technique depending on the context of the application.

According to the exposed validity threats, a plan is under way to conduct replications of this study considering other business process mining techniques, as well as different dependent variables in the Bender method.

Acknowledgement

This work was supported by the FPU Program from the Spanish Ministry of Education and the R&D projects ALTAMIRA (JCCM, PII2I09-0106-2463), PEGASO/MAGO (MICIN and FEDER, TIN2009-13718-C02-01), and MEDUSAS (CDTI (MICINN), IDI-20090557).

References

[1] Aalst, W.M.P.v.d., et al., Workflow mining: a survey of issues and approaches. Data Knowl. Eng., 2003. 47(2): p. 237-267.

[2] Asencio, A., et al., Relating Expectations to Automatically Recovered Design Patterns, in Proceedings of the Ninth Working Conference on Reverse Engineering (WCRE'02). 2002, IEEE Computer Society. p. 87.

[3] Bender, R., Quantitative Risk Assessment in Epidemiological Studies Investigatin Threshold Effects. Biometrical Journal, 1999. 41(3): p. 305-319.

[4] Benlarbi, S., et al., Thresholds for Object-Oriented Measures. Institute for Information Technology, National Research Council Canada, 2000.

[5] Biermann, A.W. and J.A. Feldman, On the Synthesis of Finite-State Machines from Samples of Their Behavior. IEEE Trans. Comput., 1972. 21(6): p. 592-597.

[6] Brereton, P., et al., Using a protocol template for case study planning, in Evaluation and Assessment in Software Engineering (EASE'08). 2008: Bari, Italia. p. 1-8.

[7] Cardoso, J., Process control-flow complexity metric: An empirical validation. SCC '06: Proceedings of the IEEE International Conference on Services Computing, 2006: p. 167--173.

[8] Castellanos, M., et al., Business Process Intelligence, in Handbook of Research on Business Process Modeling, J. J. Cardoso and W.M.P. van der Aalst, Editors. 2009, Idea Group Inc. p. 456-480.

[9] Cohen, J., Applied multiple regression/correlation analysis for the behavioral sciences. 2003: Lawrence Erlbaum.

[10] Coleman, D., B. Lowther, and P. Oman, The Application of Software Maintainability Models in Industrial Software Systems. Journal of Systems and Software, 1995. 29(1): p. 3-16.

[11] Chidamber, S.R. and C.F. Kemerer, A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, 1994. 20(6): p. 476-493.

[12] Davis, J. and M. Goadrich, The relationship between Precision-Recall and ROC curves, in Proceedings of the 23rd international conference on Machine learning. 2006, ACM: Pittsburgh, Pennsylvania. p. 233-240.

[13] Frakes, W.B. and T.P. Pole, An Empirical Study of Representation Methods for Reusable Software Components. IEEE Trans. Softw. Eng., 1994. 20(8): p. 617-630.

[14] García, F., et al., Towards a Consistent Terminology for Software Measurement. Information and Software Technology, 2005. 48: p. pg 631-644.

[15] Garcia, V.C., et al., From Specification to Experimentation: A Software Component Search Engine Architecture, in 9th International Symposium on Component-Based Software Engineering (CBSE 2006). 2006, Springer-Verlag: Västerås, Sweden. p. 82-97.

[16] Heuvel, W.-J.v.d., Aligning Modern Business Processes and Legacy Systems: A Component-Based Perspective (Cooperative Information Systems). 2006: The MIT Press.

[17] ISO/IEC, ISO/IEC DIS 19506. Knowledge Discovery Meta-model (KDM), v1.1 (Architecture-Driven Modernization). http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2=080&ics3=&csnumber=32625. 2009, ISO/IEC. p. 302.

[18] Jeston, J., J. Nelis, and T. Davenport, Business Process Management: Practical Guidelines to Successful Implementations. 2nd ed. 2008, NV, USA: Butterworth-Heinemann (Elsevier Ltd.). 469.

[19] Koskinen, J., et al. Software Modernization Decision Criteria: An Empirical Study. in European Conference on Software Maintenance and Reengineering. 2005: IEEE Computer Society.

[20] Lou, J.-G., et al., Mining program workflow from interleaved traces, in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. 2010, ACM: Washington, DC, USA. p. 613-622.

[21] Lucrédio, D., R.P.M. Fortes, and J. Whittle, MOOGLE: A Model Search Engine, in 11th international conference on Model Driven Engineering Languages and Systems. 2008, Springer-Verlag: Toulouse, France. p. 296-310.

[22] McCabe, T.J., A Complexity Measure. IEEE Transactions on Software Engineering, 1976. SE-2(4): p. 308-320.

[23] Mendling, J., Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness. 2008: Springer Publishing Company, Incorporated.

[24] Miller, G.A., The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychological Review, 1956. 63(2): p. 81--97.

[25] Nakache, D., E. Metais, and J.F. Timsit, Evaluation and NLP, in Database and Expert Systems Applications, K.V. Andersen, J. Debenham, and R. Wagner, Editors. 2005, Springer Berlin / Heidelberg. p. 626-632.

[26] Nejmeh, B.A., NPATH: a Measure of Execution Path Complexity and its Applications. ACM, 1988. 31(2): p. 188-200.

[27] OMG, QVT. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. http://www.omg.org/spec/QVT/1.0/PDF. 2008, OMG.

[28] OMG, Business Process Model and Notation (BPMN) 2.0. 2009, Object Management Group. p. 496.

[29] Pérez-Castillo, R., I.G.-R. de Guzmán, and M. Piattini, Business Process Archeology using MARBLE. Information and Software Technology, 2011. 53: p. 1023–1044.

[30] Pérez-Castillo, R., et al., Business Process Patterns for Software Archeology, in 25th Annual ACM Symposium on Applied Computing (SAC'10). 2010, ACM: Sierre, Switzerland. p. 165-166.

[31] Pérez-Castillo, R., I. García-Rodríguez de Guzmán, and M. Piattini, Implementing Business Process Recovery Patterns through QVT Transformations, in International Conference on Model Transformation (ICMT'10). 2010, Springer-Verlag: Málaga, Spain. p. 168-183.

[32] Pérez-Castillo, R., et al., A Case Study on Business Process Recovery using an E-Government System. Software Practice & Experience Journal, 2011: p. In Press.

[33] Pérez-Castillo, R., et al., An Empirical Comparison of Static and Dynamic Business Process Mining, in 26th Annual ACM Symposium on Applied Computing (SAC'11). 2011, ACM: TaiChung, Taiwan. p. 269-276.

[34] Philippow, I., et al., An approach for reverse engineering of design patterns. Software and Systems Modeling, 2005. 4(1): p. 55-70.

[35] Povzner, A., et al., Autograph: automatically extracting workflow file signatures. SIGOPS Oper. Syst. Rev., 2009. 43(1): p. 76-83.

[36] Pradel, M., P. Bichsel, and T.R. Gross, A Framework for the Evaluation of Specification Miners Based on Finite State Machines, in 26th IEEE International Conference on Software Maintenance (ICSM'10). 2010: Timi oara, Romania. p. In Press.

[37] Raghavan, V., P. Bollmann, and G.S. Jung, A critical investigation of recall and precision as measures of retrieval system performance. ACM Trans. Inf. Syst., 1989. 7(3): p. 205-229.

[38] Runeson, P. and M. Höst, Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Eng., 2009. 14(2): p. 131-164.

[39] Sánchez González, L., et al., Measurement in Business Processes: a Systematic Review. Business process Management Journal, 2010. 16(1): p. 114-134.

[40] Shatnawi, R., An Investigation of CK Metrics Thresholds. ISSRE Supplementary Conference Proceedings, 2006.

[41] van der Aalst, W., H. Reijers, and A. Weijters, Business Process Mining: An Industrial Application. Information Systems, 2007. 32(5): p. 713-732.

[42] Van Rijsbergen, C.J., Information Retrieval. 1979: Butterworths.

[43] Weske, M., Business Process Management: Concepts, Languages, Architectures. 2007, Leipzig, Alemania: Springer-Verlag Berlin Heidelberg. 368.

[44] Ye, Y. and G. Fischer, Supporting reuse by delivering task-relevant and personalized information, in Proceedings of the 24th International Conference on Software Engineering. 2002, ACM: Orlando, Florida. p. 513-523.

[45] Yeh, D., et al., An Empirical Study of a Reverse Engineering Method for Aggregation Relationship Based on Operation Propagation, in Proceedings of the 29th Annual International Computer Software and Applications Conference - Volume 01. 2005, IEEE Computer Society. p. 95-100.

5.2. Meta-Analysis

This section combines the results obtained from all the case studies by applying meta-analysis techniques. The objective is the integration of results in order to obtain strengthened conclusions.

The content of this section is a paper published in the Journal of Systems and Software. This journal had an impact factor of 1.277 in 2010, according to Thomson's Science Citation Index (SCI). The reference of this paper is:*Pérez-Castillo, R., J.A. Cruz-Lemus, I. García-Rodríguez de Guzmán, and M. Piattini, A Family of Case Studies on Business Process Mining using MARBLE. Journal of Systems and Software, 2012. 86(6): p. 1370–1385.*

The Journal of Systems and Software 85 (2012) 1370–1385

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A family of case studies on business process mining using MARBLE

Ricardo Pérez-Castillo*, José A. Cruz-Lemus, Ignacio García-Rodríguez de Guzmán, Mario Piattini

Instituto de Tecnologías y Sistemas de Información at University of Castilla-La Mancha, Spain

ARTICLE INFO

Article history:
Received 10 June 2011
Received in revised form
22 November 2011
Accepted 9 January 2012
Available online 24 January 2012

Keywords:
Business process
BPMN
Static analysis
ADM
KDM
Case study
Meta-analysis

ABSTRACT

Business processes, most of which are automated by information systems, have become a key asset in organizations. Unfortunately, uncontrolled maintenance implies that information systems age overtime until they need to be modernized. During software modernization, ageing systems cannot be entirely discarded because they gradually embed meaningful business knowledge, which is not present in any other artifact. This paper presents a technique for recovering business processes from legacy systems in order to preserve that knowledge. The technique statically analyzes source code and generates a code model, which is later transformed by pattern matching into a business process model. This technique has been validated over a two-year period in several industrial modernization projects. This paper reports the results of a family of case studies that were performed to empirically validate the technique using analysis and meta-analysis techniques. The family of case studies demonstrates that the technique is feasible in terms of effectiveness and efficiency.

© 2012 Elsevier Inc. All rights reserved.

1. Introduction

Business processes are increasingly becoming an essential asset for organizations since they create value for customers and reflect all operations of an organization (Chang, 2006). Business processes depict a set of related activities that are performed in an organization and that together realize a business goal (Weske, 2007). These descriptions provide a means to map business objectives regarding how to best carry out operations. Organizations adopt business process management through their enterprise information systems. Business processes therefore also assist at the beginning of the development of information systems to automate all those activities needed to achieve the business objectives (Heuvel, 2006).

However, enterprise information systems are not static entities, since all software ages as a result of uncontrolled maintenance over time (Visaggio, 2001). The immediate effect of software ageing is that information systems become *Legacy* Information Systems (LIS) and must therefore be replaced with a new, improved system. This activity is known as *software modernization*. Unfortunately, when organizations modernize their legacy information systems, the organizations' business processes do not reflect all changes that have occurred during the maintenance of their systems. Indeed, very often the business process documentation is neither updated nor documented at all (Lehman, 1984) Thereby, the business process models considered by the organization are not aligned with the actual business process that is executed through legacy information systems (Heuvel, 2006). Owing to the alignment problem, a legacy information system cannot be entirely discarded during its modernization since it might contain a considerable amount of latent meaningful business knowledge (Koskinen, 2004). This knowledge is embedded in the system as a consequence of modifications over time and it might not, therefore, be present anywhere else. As a result, all embedded business processes must be explicitly recovered in order to preserve this meaningful asset in the modernized information systems (Paradauskas and Laurikaitis, 2006). The recovered business processes can then be used by organizations in two ways: (i) to provide business experts with a better understanding of the real, current operation of the organization; and (ii) if necessary, to develop the new improved system in order to mitigate the effects caused by the software ageing. The evolved system will thus support the current business processes and will also improve the ROI (*Return Of Investment*) of the legacy system, since it extends its lifespan.

This paper deals with the problem of business process mining from legacy information systems in order to address the challenge of business knowledge preservation. The paper provides a semi-automatic technique for recovering business processes from legacy information systems. A supporting tool is provided to automate and facilitate the adoption of the technique. The proposal consists of a reverse engineering technique that follows the model-driven

* Corresponding author.
E-mail addresses: ricardo.pdelcastillo@uclm.es (R. Pérez-Castillo), joseantonio.cruz@uclm.es (J.A. Cruz-Lemus), ignacio.grodriguez@uclm.es (I.G.-R. de Guzmán), mario.piattini@uclm.es (M. Piattini).

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2012.01.022

development principles. The technique therefore depicts a set of models and the transformations between them to obtain a set of preliminary business processes. The technique is aided by the manual post-intervention of business experts who refine the business processes obtained (e.g., by adding manual tasks that are not supported by information systems). The technique is therefore considered to be semi-automatic. Despite this, the technique is better than that of business experts redesigning business processes from scratch, since that solution is more time-consuming (Mutschler and Reichert, in press) and error-prone than semi-automatic techniques (Weerakkody and Currie, 2003). Furthermore, the manual business process redesign by business experts may ignore some business knowledge embedded in LISs.

The proposed technique is framed in MARBLE, a generic ADM-based framework to support business process mining, which was presented in a previous work (Pérez-Castillo et al., 2009a). ADM (Architecture-Driven Modernization) as defined by the OMG (Object Management Group) (OMG, 2009a), is a standard that advocates carrying out reengineering processes by considering model-driven development principles. The objective of MARBLE is to provide the highest abstraction level during the reverse engineering stage of ADM, i.e., the business knowledge that LISs depict. The MARBLE framework thus provides a solution to meet the demands detected by Khusidman and Ulrich (2007). It is also aligned with the SOA (Service-Oriented Architecture) research agenda developed by the SEI (Software Engineering Institute) (Lewis et al., 2010), which reports that business process recovery is needed to modernize LISs towards SOA systems.

All of the techniques framed in MARBLE are characterized by two key factors: (i) the set of knowledge sources, i.e., the software artifacts (e.g. source code, databases, user interfaces, etc.) used for recovering business processes; and (ii) the kind of reverse engineering technique (e.g. static or dynamic analysis of source code, program slicing, etc.) used to extract meaningful business knowledge. The proposed technique (i) considers legacy object-oriented code as the source of knowledge; and (ii) uses static (i.e. syntactical) analysis of source code as a reverse engineering technique. It is therefore possible to state that the technique supports static business process mining.

The proposed technique has been empirically validated using the supporting tool over the last two years by means of five industrial case studies involving five real-life LISs. In order to improve their rigor and validity, the case studies were conducted by following the formal protocol for case studies proposed by Runeson and Höst (2009). Each case study provided meaningful information concerning the effectiveness and efficiency of the proposed technique. This paper groups all that information and provides the meta-analysis of the results obtained for all case studies in order to attain strengthened conclusions, to evaluate relationships between the results obtained in each study, and to assess their effectiveness and efficiency measures. The empirical validation demonstrates that the technique enables the recovery of business processes with adequate accuracy levels, and that it is possible to do this in a scalable manner for large LISs.

The remainder of this paper is organized as follows. Section 2 summarizes the related work. Section 3 presents the proposed technique in detail. Section 4 presents the planning and execution of the family of case studies. Finally, Section 5 discusses the main conclusions attained from the meta-analysis, along with our future work.

2. Related work

Business knowledge preservation is not a new problem, and reverse engineering applied to the recovery of business processes from LISs has been frequently addressed by both business experts and software academics. Some works address business knowledge recovery from LISs by statically analyzing various software artifacts. For instance, Ghose et al. (2007) use text-based queries executed in documentation to extract business knowledge. This proposal was only validated through an example. System databases are other artifacts that are used as input in business knowledge recovery based on static analysis. Paradauskas and Laurikaitis (2006) recover business knowledge by means of the inspection of the data stored in databases together with legacy application code. The authors of this work validated their proposal through the development of a controlled example. Wang et al. (2004) present a framework for business rule extraction from large LISs based on static program slicing. Program slicing is a reverse engineering technique consisting of decomposing the program into slices according to certain criteria (e.g. fragments of source code that uses a specific program variable). However, the framework is not able to represent the business knowledge recovered in an understandable and standard manner. This work was validated by means of a large complex financial legacy system, but this validation did not follow a formal protocol and no meta-analysis techniques were used to analyze the results in combination. do Nascimento et al. (2009) present a method for rewriting LISs based on business process management, but it provides a framework that consists of a manual business process recovery. None of these works follow model-driven development principles. Those works that present ad hoc solutions therefore have some formalization and automation problems. The validation of this proposal is based on model conformance checking through the π-*calculus* tool (Milner et al., 1992).

Zou et al. (2004) developed a model-driven framework based on a set of heuristic rules that syntactically analyze the source code and extract business processes. Although this work follows some model-driven development principles, it does not consider the ADM standard. This proposal was validated by means of a case study concerning an e-commerce system, but no formal protocol to conduct the case study was used. Another work which statically analyzes a database as the input artifact is provided by Pérez-Castillo et al. (2009b). This work proposes a model-driven reengineering framework to extract business logic from relational database schemas. This proposal was validated by means of a case study in which the framework was applied to a real legacy relational database of a research institute.

The only reverse engineering technique used in all of the aforementioned works is that of static analysis, which has the inconvenience that a lot of business knowledge is lost since it ignores all runtime knowledge. Dynamic analysis solves this problem since it facilitates consideration of the information derived by system execution (Cornelissen et al., 2009). For instance, Eisenbarth et al. (2003) present a business feature location technique based on dynamic analysis. This proposal neither follows model-driven principles nor extracts business knowledge according to any business process notation, and demonstrates its applicability by means of an example. Cai et al. (2009) propose an approach that combines the requirement reacquisition with a dynamic and static analysis technique to extract complex business processes that are triggered by external actors. This work was also validated by means of a non-formal case study. In addition, Di Francescomarino et al. (2009) recover business processes by dynamically analyzing the Web application GUI-forms, which are executed during a user's navigation. This work additionally provides a clustering algorithm for minimizing business processes obtained. However, this proposal is not aligned with model-driven development principles either. The validation in this case was made through a case study with an e-commerce system which was conducted in a non-formal way.

Other works addressing the dynamic approach provide process mining techniques that register event logs focusing on Process-Aware Information Systems (PAIS), i.e., process management systems such as Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM). The nature of these systems (and particularly their process-awareness) facilitates the registration of events throughout process execution. Event logs depict the sequence of business process activities executed, and can be used to dynamically discover the current business processes. A considerable amount of work follows this approach. For example, van der Aalst et al. (2007) provides different business process mining algorithms to discover business process form event logs which were validated throughout a large industrial case study by applying different algorithms. Gunther et al. (2007) provides a generic import framework with which for obtaining event logs from different kinds of process-aware information systems. Ingvaldsen et al. (2008) focus on ERP systems to obtain event logs from the SAP's transaction data logs. However, neither work follows model-driven development principles. These two proposals were validated by applying them to real systems but both studies did not follow a formal protocol.

Dynamic analysis is more powerful and accurate than static analysis. The main drawback of static analysis is that it only focuses on static information retrieved by syntactically analyzing the existing source code or any other artifact. Moreover, there is much valuable knowledge that is known only at run time, for which dynamic analysis are more suitable. However, the applicability of dynamic analysis has some limitations since it usually requires source code modifications in order to aggregate traces to collect system execution information in an appropriate way (Canfora et al., 2011). In practice, source code modifications are not always possible since legacy information systems can be in the production stage. Another point of controversy related to the usage of dynamic analysis techniques in realistic environments is that it might be thought that a company or organization would not accept the use of an automatically modified version of its information system. For this reason, dynamic analysis techniques should ensure that the source code instrumentation is carried out without affecting the behavior of the original information system, i.e. in a non-invasive way. The main advantage of static analysis is its efficiency with regard to dynamic one.

Our proposal is framed in MARBLE, and is thus based on the ADM standard. It provides a semi-automatic reverse engineering technique based on static analysis of source code. The proposal is validated by a set of case studies as well as a meta-analysis study applied with the obtained results. Table 1 shows the comparison between the proposed technique and the other related work.

Table 1
Comparison of work related to business process recovery.

	Our proposed technique*	Ghose et al. (2007)	Paradauskas and Laurikaitis (2006)	Wang et al. (2004)	do Nascimento et al. (2009)	Zou et al. (2004)	Pérez-Castillo et al. (2009b)	Eisenbarth et al. (2003)	Cai et al. (2009)	Di Francescomarino et al. (2009)	van der Aalst et al. (2007)	Gunther et al. (2007)	Ingvaldsen et al. (2008)
Static analysis	×	×	×	×		×	×	×	×	×	×	×	×
Dynamic analysis													
Recovery from event logs	×					×	×						
Model-driven development principles													
ADM	×												

3. Static business process mining

The proposed business process recovery technique is framed in MARBLE. On the one hand, MARBLE depicts a general-purpose and extensible ADM-based framework for recovering business processes (cf. Section 3.1). MARBLE establishes the guidelines for various techniques whose objective is the same: retrieve business process models from legacy software artifacts. All particular MARBLE-based techniques can be considered as an instance of MARBLE and the commonalities of such techniques are: (i) the information is collected into and is used from standard KDM (Knowledge Discovery Metamodel) repositories and (ii) the information of KDM repositories is used to retrieve business process models.

On the other hand, the certain MARBLE-based technique presented in this paper establishes the three specific transformations needed to recover the business process (cf. Section 3.2).

MARBLE-based techniques consist of three parts: (i) legacy software artifacts used to retrieve business process insights (e.g., source code, databases, etc.); (ii) technique to retrieve information from legacy software artifacts (e.g., static analysis vs. dynamic analysis); and (iii) the transformation or technique from KDM repository to business process models. The paper proposes a technique based on static analysis of object-oriented source code and manual post-intervention by business experts as the further valuable sources of knowledge.

3.1. MARBLE

Reengineering has normally been used to obtain new improved versions of aged LISs and, according to Chikofsky and Cross (1990), consists of three stages: reverse engineering, restructuring, and forward engineering. However, reengineering projects rarely reach the business abstraction level – they typically concentrate on the level of system design (Khusidman and Ulrich, 2007). According to Sneed (2005), reengineering usually lacks for formalism and standardization. Indeed, the majority of reengineering projects are usually carried out in an ad hoc manner. ADM helps to solve the formalism and standardization problems of reengineering. ADM-based processes deal with all artifacts involved in reengineering as models that conform to specific metamodels, and these processes also establish model transformations between models to deal with different abstraction levels throughout the three reengineering stages (Newcomb, 2005).

MARBLE (*Modernization Approach for Recovering Business processes from Legacy Systems*) (Pérez-Castillo et al., 2009a) is an ADM-based generic framework that facilitates business process recovery. All particular techniques framed in MARBLE specify the path of model transformations (cf. Section 3.2) needed to obtain each model in a specific level from a previous one. MARBLE defines four abstraction levels related to four different kinds of models: L0 to L3.

- *L0. Legacy information system.* This level represents the entire LIS that exists in the real world as a set of interrelated software artifacts: source code, user interfaces, databases, documentation, etc.
- *L1. Legacy information system models.* This level contains a set of models that can represent one or more software artifacts of the LIS at L0. These models are considered to be platform-specific models (PSM) since they represent different views or concerns of the systems from a technological point of view at a lower abstraction level.
- *L2. KDM model.* This level contains a single model that integrates all models of the previous L1 level. This level represents the entire LIS from a platform-independent model (PIM) at an intermediate abstraction level. MARBLE uses the KDM (Knowledge Discovery Metamodel) standard to represent this model. KDM is an ADM specification proposed by the OMG and has been recognized as the ISO 19506 standard (ISO/IEC, 2009). KDM provides a common repository structure that makes it possible to exchange information about all legacy software artifacts, and can be compared with the Unified Modeling Language (UML) standard; while UML is used to generate new code in a top-down manner, an ADM process involving KDM starts from the existing code and builds a higher level model in a bottom-up manner (Pérez-Castillo et al., in press).
- *L3. Business process models.* This level corresponds to models that represent the business processes recovered from the LIS. The models in this level are considered to be computer-independent models (CIM) since they depict a business view of the system at the highest abstraction level. In order to represent this kind of models, MARBLE uses the BPMN (Business Process Modeling and Notation) standard metamodel (OMG, 2009b), since it offers a well-known graphical notation that is easily understood by both system analysts and business experts.

3.2. Proposed technique to recover business processes

The proposed business process recovery technique is framed in MARBLE, which is also automated by a supporting tool (Alarcos Research Group MARBLE Tool, 2011). The technique defines three model transformations and progressively presents a working example for explaining these transformations.

3.2.1. L0-to-L1 transformation

The first transformation takes into account legacy object-oriented code as the sole software artifact, since it is the artifact that embeds most business knowledge. This transformation thus obtains a code model that represents the source code, considering technological details. Indeed, this transformation must be tuned for each programming language, e.g. it is developed for Java-based systems. This transformation therefore consists of the static analysis of Java code files, which is carried out by means of a syntactical parser. As a result of the analysis, it builds code models according to the Java metamodel, which represents abstract syntax trees of code files.

An example concerning an information system supporting the product shipping of a reseller organization is now introduced to illustrate how the transformation works. This example focuses on a certain piece of source code: the '*ResellerControler*' class of the '*domain*' package (see Fig. 1, left). This class contains four methods that support four key functionalities of the system: (i) '*receiveOrder*' manages customers' order requests; (ii) '*checkInventory*' checks whether the products needed to fulfill a specific order are in stock; (iii) '*sendProducts*' manages the dispatch of the products ordered; and finally (iv) '*sendInvoice*' generates and sends the invoice to the customer.

According to the example, the L0-to-L1 transformation takes the Java source file (see Fig. 1, left) and obtains the abstract syntax tree that represents the java code model at L1 (see Fig. 1, right). The parser obtains a *CompilationUnit* element for each of the java source files analyzed, and then adds the *PackageDeclaration* and *ImportDeclaration* elements. The parser then generates a *ClassOrInterfaceDeclaration* element for the class, which contains a *ClassOrInterfaceBodyDeclaration* element for each method. Each method is represented through a *MethodDeclaration* with *ResultType*, *MethodDeclarator* and *Block* of *Statement* elements. The *Statement* element is, in turn, specialized into several kinds of elements: *ReturnStatement*, *IfStatement*, *Expression*, and so on. The right-hand side of Fig. 1 shows the elements used to represent the business logic implemented in the '*receiveOrder*' method.

3.2.2. L1-to-L2 transformation

The L1-to-L2 transformation is in charge of the transformation of the PSM models at L1 into a single PIM model at L2 according to the KDM metamodel. The KDM metamodel consists of several metamodel packages which are interrelated and organized into four abstraction layers. Each package defines a set of metaclasses with which to model a different view or concern of LISs. The KDM model at L2 considers the *code* and *action* KDM packages of which the *program element* layer of the KDM metamodel is composed. The *code* and *action* KDM packages provide a language-independent intermediate representation for various constructs determined by common programming languages. This transformation uses only these packages because legacy source code is the only artifact considered at L0. However, owing to the KDM representation, this information might, in the future, be integrated in a standardized manner with new valuable knowledge related to other software artifacts.

Fig. 1. An example of the L0-to-L1 transformation for a product shipping system.

In order to establish the model transformation between the code model at L1 and the KDM code model at L2, the technique uses the QVT (Query/Views/Transformation) language to implement the transformation. Owing to the fact that the structures of the Java metamodel (L1) and KDM code-action metamodel (L2) are very similar, the implementation of this transformation is almost immediate when using QVT (a declarative language), since it usually only needs to rename the metaclasses of both metamodels.

To continue with the example, Fig. 2 shows the KDM model obtained after the execution of the proposed QVT transformation for the product shipping system. The KDM model contains a *Package* instance named '*domain*' with a nested *CompilationUnit* instance named '*Reseller*'. The *CompilationUnit* is obtained from the *Class* instance of the Java code model at L1. The '*Reseller*'*CompilationUnit* instance contains an instance of the *CallableUnit* metaclass for each Java method at L1. Each *CallableUnit* instance is also defined by means of different *CodeElement* and *ActionElement* instances. For example, the '*receiveOrder*' method is modeled as a *CallableUnit* containing (see Fig. 2): (i) an *EntryFlow* instance that defines the first KDM action in the unit (the first Java statement, since a model's sequentiality of actions is not clearly defined); (ii) a *Signature* instance that defines the parameters of the unit; and finally (iii) an *ActionElement* instance that represents the *if* statement in the Java method. The remaining *CallableUnit* instances have a similar structure.

3.2.3. L2-to-L3 transformation

The L2-to-L3 transformation is the last transformation and obtains business process models at L3 from the previous KDM model at L2. This transformation uses the metamodel of the BPMN standard (OMG, 2009b) to represent the business process models. Fig. 3 shows the BPMN metamodel, which enables the representation of business process diagrams as instances of the *BusinessProcessDiagram* metaclass. Each diagram can involve four kinds of elements: (i) flow object elements such as the *Activity*, *Task*, *Event* and *Gateway* metaclasses; (ii) connecting object elements such as *SequenceFlow*, *MessageFlow* and *Association*; (iii) artifact elements such as the *DataObject*, *group* and *Annotation* metaclasses; and (iv) swim lane elements which are used to group elements such as *Pool* and *Lane*.

This transformation consists of two steps: (i) a model transformation that obtains a set of preliminary business process models; and (ii) manual intervention by business experts that modifies the business processes obtained in order to improve them. Firstly, the model transformation establishes a set of business patterns (Pérez-Castillo et al., 2010a), which define which pieces of the source code (represented in a KDM code model) will be transformed into specific structures of a business process. Table 2 summarizes the set of proposed business patterns and briefly explains each pattern in natural language. The transformation is implemented through a set of QVT relations (Pérez-Castillo et al., 2010b), which support the pattern matching process. Each QVT relation searches for instances of the source structures defined by each pattern, and the QVT relations then enforce the creation (in the business process model) of instances of the target structures of the pattern for each input instance found in the KDM model.

Owing to space limitations, Fig. 4 shows only one QVT relation as an example (the *CallableUnit2Task* relation), although the complete transformation is available online (Pérez-Castillo, 2011). The *CallableUnit2Task* implements the *P2.Sequence* pattern, and transforms all *CallableUnit* instances (e.g. java methods) into *Task* instances, and invocations between them into sequence flows. This relation follows the well-known "*a callable unit-atask*" approach proposed by Zou et al. (2004). There are other choices that

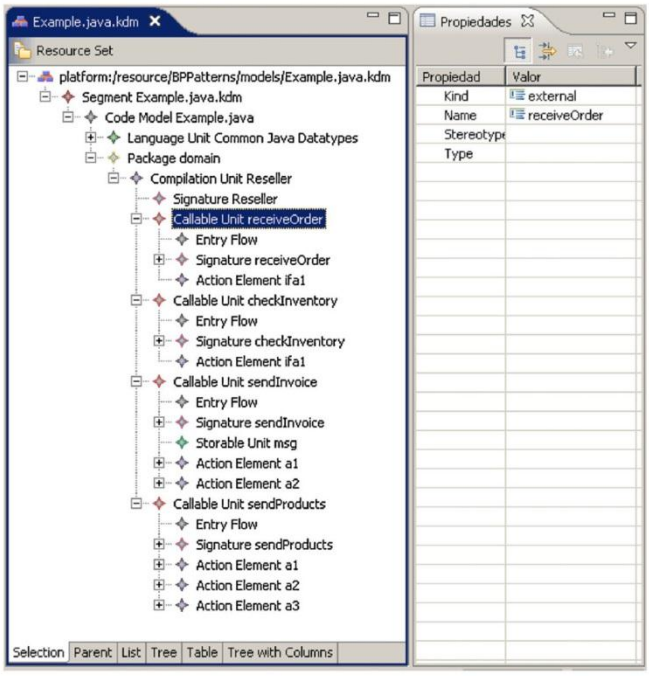


Fig. 2. The KDM model obtained after the proposed QVT transformation for the product shipping system.

transform methods and their respective calls into subprocess-task relationships (i.e. composite tasks). However, the "*a callable unit-a-task*" approach is better to solve problems regarding the method granularity, since it can treat all methods in the same manner and then, it can discard fine-grained methods using heuristics like removing the 'getters/setters' methods.

Finally, from the where clause, the *CallableUnit2Task* relation calls other relations that depend on the tasks created. For instance,

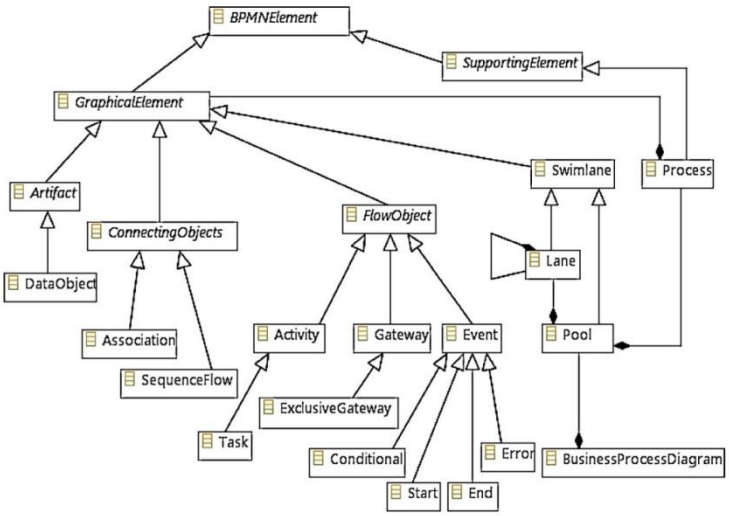
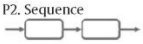
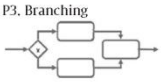
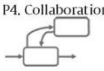
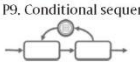
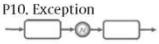


Fig. 3. Overview of the BPMN metamodel.

Table 2
Summary of the set of business patterns.

Pattern	Description
P1. BPD Skeleton	This pattern creates the root structure of the BP model. It creates a BP diagram for each KDM code model. Also, it builds a pool element with a nested process in the BP diagram for each package of the KDM code model
P2. Sequence 	This pattern takes any callable piece of code from the KDM code model and maps them into tasks in the BP diagram. The sequence of calls to callable units is transformed into a set of sequence flows in the same order between the tasks built from the callable unit respectively
P3. Branching 	This pattern transforms each conditional jump of the source code that has two mutually exclusive choices into an exclusive gateway and two different sequence flows in the BP model. Typically those exclusive conditional branches are related to the if . . . then . . . else or switch clauses in several programming languages. The exclusive gateway represents the condition that is evaluated and the two sequence flows represent two conditional transitions that depend on the value (true or false) of the evaluation
P4. Collaboration 	Each call to external callable unit (i.e. API libraries or external components outside the legacy system) is transformed into an auxiliary task as well as two sequence flows: the first from the source task to the auxiliary task and the second returning to the source task
P5. Data input	This pattern builds a data object in the BP model for each input data within a callable unit in the KDM code model. Also, this pattern builds an association between the data objects and the task previously built from the callable unit. This pattern only considers as input data the parameters or arguments of the callable unit, but it does not consider the auxiliary variables within the callable unit
P6. Data output	Each piece of output data involved in a callable unit is transformed by means of this pattern into a data object as well as an association from the task (built from the callable unit) to the data object. This pattern excludes as output data the auxiliary and intermediate data in the body of the callable unit. The output data is the data returned by the callable unit or external data related to databases or files
P7. Start	The task building from the callable unit that starts the execution of any program or application of the legacy system is considered the initial task. A start event is built into the BP diagram and a sequence flow from this event to the initial task is also created
P8. Implicit termination	This pattern builds an end event in the BP model. Then, it creates sequence flows from 'end task' and those flows merge in the end event. A task is considered an 'end task' if this task does not have any outgoing sequence flow
P9. Conditional sequence 	This pattern transforms each conditional call into a sequence flow fired under a conditional intermediate event through to the task related to the callable unit. This pattern makes it possible to create arbitrary cycles in the BP diagram
P10. Exception 	Each call to callable unit under any exception is transformed into a task for the piece of source code that handles the exception as well as a sequence flow fired under an error intermediate event. Indeed, this pattern can be understood as a specialization of the previous pattern P9

the *WritesStorableUnit2DataObject* relation that implements the *P6.Data Output* pattern is called for each piece of data written by the callable unit.

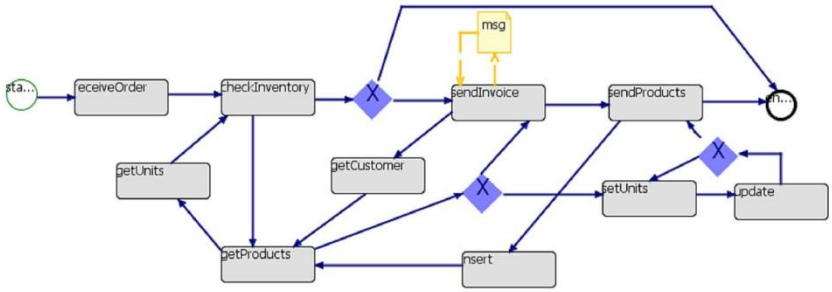
Secondly, the L2-to-L3 transformation can be additionally aided by the manual post-intervention of business experts to refine the business processes obtained. This procedure is necessary in at least two cases: (i) when the technique has recovered some tasks related to the technical nature of legacy source code and these must be removed; and (ii) when there are manual business tasks that are not supported by information systems, and which must be added. It consequently proposes a semi-automatic technique. This could be considered as a critical point of our proposal, since it requires additional manual intervention by experts. However, business process redesign from scratch is a more time-consuming and difficult option than our proposal, which provides an initial meaningful understanding of current business processes. From an effort/cost viewpoint, any supporting tool during redesign increases the ability to redesign business processes, which is a factor to reduce the redesign effort (Mutschler and Reichert, in press).

```
relation CallableUnit2Task {
   xName : String;
   checkonlydomainkdm m : code::CallableUnit {
       name = xName
   };
   enforcedomainbpmnpr : bpmn::Process {
       GraphicalElements = t :bpmn::Task {
           Name = xName,
           Status = bpmn::StatusType::None,
       }
   };
   where {
       ...
       m.codeElement->forAll (a: AbstractCodeElement | a.oclAsType
       (ActionElement).actionRelation->forAll (w:AbstractActionRelationship |
       (w.oclIsKindOf(Writes)) andw.oclAsType(Writes).to.oclIsKindOf
       (StorableUnit) implies WritesStorableUnit2DataObject (w, t, pr)));
       ...
   }
}
```

Fig. 4. A piece of QVT implementation of the L2-to-L3 transformation.

To continue with the example, at the beginning the L2-to-L3 transformation obtains a first sketch of a business process diagram from the KDM model by means of the QVT transformation. Fig. 5(A) shows the graphical representation of the business process model obtained during the QVT transformation. This model contains 10 tasks in total, although only 4 tasks are related to the four *CallableUnit* instances in the KDM model (see Fig. 2), which are obtained by applying the '*P2.Sequence*' pattern. The gateways are also created by applying the '*P3. Branching*' pattern. Moreover, 6 other tasks are also obtained by applying the '*P4.Collaboration*' pattern (see Table 2). This pattern is applied to the three *ActionElement* instances of the *CallableUnit* instance named '*sendProducts*' in the KDM model (see Fig. 2). These action elements represent calls to methods, which are not defined in the same Java source file (e.g. '*update*', '*setUnits*'

(A) Preliminary business process model obtained by means of the QVT transformation



(B) Business process model after manual intervention by business experts

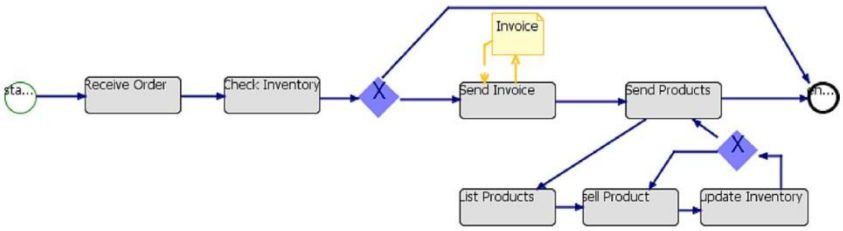


Fig. 5. An example of business process models obtained for a product shipping system.

and '*insert*'), and these calls are thus represented as external calls, which are transformed into six tasks with a round-trip sequence flow from the previous tasks.

Fig. 5(B) shows the business process model refined by business experts. There are three principal improvements:

- The three tasks obtained from external calls are removed, since the business experts consider that these tasks do not represent any of the organization's business activities, i.e. they are related to the technical dimension of the system.

```
Remove    ['getCustomer'];
Remove    ['getUnits'];
Remove    ['insert'];
```

- A gateway is added to merge the two branches opened after the first gateway.

```
Add[GATEWAY['sendProducts','gateway(CheckInventory)', END]];
```

- The remaining tasks and data objects are renamed by the business experts in order to fit the names to the organization's business activities.

```
Rename    ['receiveOrder','Receive Order'];
Rename    ['checkInventory','Check Inventory'];
Rename    ['sendProducts','Send Products'];
Rename    ['sendInvoice','Send Invoice'];
Rename    ['getProducts','List Products'];
Rename    ['update','Update Inventory'];
Rename    ['setUnits','Sell Product'];
Rename    ['msg','Invoice'];
```

Finally, Fig. 5(B) shows the business process model that represents a part of the organization's behavior through the business process recovery from the piece of source code presented at the beginning of this example (see Fig. 1, left).

4. Family of case studies

This section presents the family of case studies carried out over the last 2 years. The family was performed in five different information systems to validate the effectiveness and efficiency of the proposal. The case studies were carried out by following the formal protocol for case studies proposed by Runeson and Höst (2009) to improve their rigor and validity. The following sections present the stages of the protocol in detail: design, case selection, execution, data collection, analysis and interpretation, and an evaluation of its validity.

4.1. Design

The *object of study* is the business process mining technique, and the *purpose of this study* is to quantitatively evaluate the specific properties of the technique as regards their effectiveness and efficiency. A qualitative research about the usage of retrieved business process models to modernize legacy information systems could be very interesting. However, the evaluation of this usage is outside of the scope of the empirical evaluation since (i) such usage of business processes may be quite different in each case and (ii) the assessment of this usage would be probably carried out through surveys involving people that possibly do not have the same expertise level about business process management, which may imply biased results.

According to the object and purpose of the study, the main research question (*MQ*) is the following: *Can the technique properly obtain business processes from legacy systems?* In this question, the adverb '*properly*' is related to the feasibility and suitability of the technique, which means, in turn, business processes must be obtained in an effective and efficient manner. In order to discover

Table 3
Case study research questions.

Id	Research question	Evaluated properties
MQ	Can the technique properly obtain business processes from legacy information systems?	Feasibility/suitability
AQ1	Can the technique obtain business processes with adequate accuracy levels?	Effectiveness
AQ2	Is the proposed technique efficient?	Efficiency

whether *MQ* is true, Table 3 shows the additional research questions (AQ) that are identified from *MQ*. *AQ1* is established in order to evaluate whether the business processes recovered faithfully represent the organization's business behavior. That is, *AQ1* aims to evaluate the effectiveness of the procedure. Moreover, the additional research question *AQ2* (see Table 3) evaluates whether the proposed business process recovery procedure obtains business processes efficiently.

To answer the research questions, the study evaluates some properties by analyzing all retrieved business processes (see Table 3). For checking these properties, some measures are collected to provide quantitative answers to the proposed research questions.

Furthermore, the study follows an *embedded* design, since each study focuses on several analysis units within each single case. The analysis units are the different source code packages of each LIS, which is the independent variable of each study (i.e., the defined measures are evaluated for each analysis unit). Each source code package is represented in a code model at L1, in a KDM model at L2, and is eventually transformed into a business process model at L3.

The usage of code packages as the minimal unit to be transformed into a business process model could provide some processes with lower accuracy levels than other solutions considering correlation data set at the beginning of the process. However, the advantage of this heuristic solution is that it does not require the initial correlation information, and it automatically provides a preliminary business processes by statically analyzing the legacy source code. Those business processes can be used as the basis to understand the entire business processes of the organization, which can be joined or split through manual post-intervention by business experts.

4.1.1. Evaluation of effectiveness

In order to evaluate the effectiveness of the technique according to *AQ1*, the study uses the *precision* and *recall* measures. These measures were designed for information retrieval scenarios, although they are usually applied to model recovery scenarios. Precision and recall measure the similarity between a mined business process *M* and a reference business process *R*. Precision indicates what proportion of *M* matches *R* (i.e., how exact *M* is), while recall indicates what proportion of *R* is present in *M* (i.e., how complete *M* is).

The study considers the *task* element as the score unit in order to apply these measures in a business process recovery scenario. It uses the opinions of business experts to discover which recovered tasks are or are not relevant. The precision measure (1) is therefore defined as the number of recovered relevant tasks divided by the total number of recovered tasks, while the recall measure (2) is defined as the number of recovered relevant tasks divided by the total number of relevant tasks. Although precision and recall are adequate, there is an inverse relationship between them. As a result, it is difficult to extract conclusions by using an isolated evaluation of these metrics. These measures are therefore usually combined into a single measure known as an *F*-measure (3), which consists of a weighted harmonic mean of both measures.

$$\text{Precision} = \frac{\{\text{recove rerelevant tasks}\}}{\{\text{recovered relevat tasks}\} + \{\text{recovered non-relevat tasks}\}} \quad (1)$$

$$\text{Recall} = \frac{\{\text{recove rerelevant tasks}\}}{\{\text{recovered relevat tasks}\} + \{\text{non-recovered relevat tasks}\}} \quad (2)$$

$$F\text{-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

Fig. 6 and Table 4 shows some example business process models to enable the reader to understand how measures work. Fig. 6(A) shows a desirable source business process model with five tasks. Fig. 6(B) shows an example business process model retrieved with

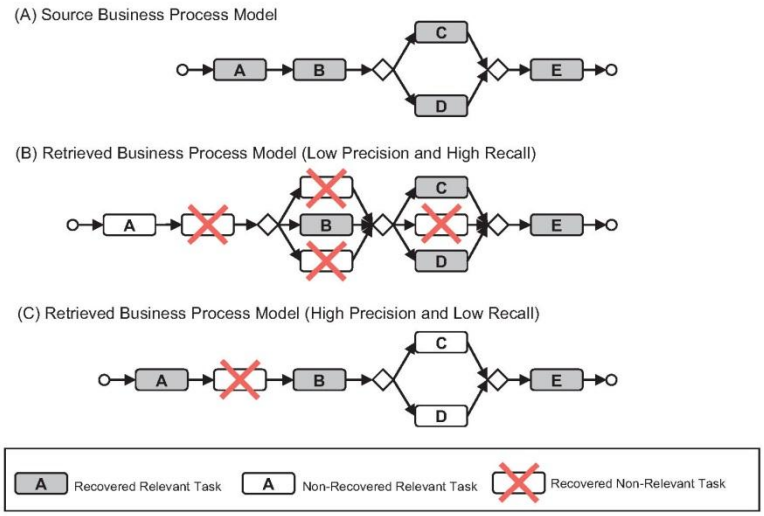


Fig. 6. Some business process models retrieved regarding a source model.

Table 4
Examples of measure values for some business process models.

Model	# Relevant tasks	# Recovered tasks	# Recovered relevant tasks	# Recovered non-relevant tasks	# Non-recovered relevant tasks	Precision	Recall	*F*-measure
A	5	5	5	0	0	1	1	1
B	5	8	4	4	1	0.5	0.8	0.61
C	5	4	3	1	2	0.75	0.6	0.67

four source tasks. One more task must thus be added by manual intervention. In addition, four other retrieved tasks are erroneous. This model presents low precision and high recall values since several non-relevant tasks were erroneously recovered (see Table 4). Finally, Fig. 6(C) provides another example business process model which was retrieved with three source tasks and one erroneous task. This model presents higher precision and lower recall than the previous model since the number of recovered non-relevant tasks is lower and the number of non-recovered relevant tasks is higher. The business process model retrieved in Fig. 6(C) is better than the model in Fig. 6(B) since the *F*-measure of (C) is higher (see Table 4) owing to the fact that the precision and recall values are more balanced (i.e., these values are closer).

Other important aspect to evaluate precision and recall is how a task is considered as a relevant task. Business experts check four conditions that should be met by the task under evaluation. The first condition specifies that the task must represent a real-life business operation within the organization. This condition is not evaluated by considering task names, since these names are inherited from legacy code and they may be biased regarding the real business activity names provided by business experts. The second condition ensures that all relevant tasks preceding the evaluated task must be recovered before the task under evaluation. In order to fulfill this condition, the predecessor tasks can be directly or indirectly connected to the task under evaluation, i.e., there could be non-relevant tasks intercalated between the evaluated task and their predecessor relevant tasks. In a similar manner, the third condition ensures that all subsequent tasks must be directly (or indirectly) recovered relevant tasks. Finally, the fourth condition specifies that all data objects related to the task under evaluation have been also recovered. The condition-based checking facilitates a final business process model after several iterations of evaluation.

4.1.2. Evaluation of efficiency

In order to evaluate the efficiency of the technique according to *AQ2*, the study employs the time spent on executing the whole transformation between MARBLE levels.

On the one hand, the *transformation time* is automatically measured in milliseconds by the tool developed. This measure is therefore analyzed with respect to the total number of elements built into each specific business process model (i.e., for each analysis unit). Both the number of elements and the transformation time are also considered as dependent variables in the study. A regression model is therefore established between the transformation time and the size of retrieved models to find out if the technique is or is not scalable to large legacy information systems.

On the other hand, the time derived by the manual intervention carried out by business experts is measured in order to know how such intervention affects to the total transformation time. This is measured in seconds for each business process model retrieved and is also considered as a dependent variable of the study.

4.2. Case selection

The five LISs were selected by means of the evaluation of four case selection criteria (*C1* to *C4*). *C1* guarantees that the LIS selected is an information system that supports an organization's business operation. This criterion discards, for example, embedded systems or real-time systems. *C2* ensures that the selected system really is an LIS. This criterion considers the amount of modifications (from the time when the system was first released) which have altered the business processes supported by the system. *C3* ensures that the system is not a *toy program*, since it defines an arbitrary threshold of 20,000 lines of source code. Finally, *C4* guarantees that the system is based on the Java platform, since the tool supporting the technique was developed for Java-based systems.

After evaluating several available systems according to the aforementioned criteria, five systems were selected to be studied. Table 5 shows the name of each system, a brief description, the kind of architecture, and the size of the source code in thousands of lines.

4.3. Execution

The execution of the study is aided by the tool, which was developed to support the proposal. The case study procedure defines the followings steps:

1. After some meetings between the staff of the candidate organizations and researchers, the set of legacy information systems is selected according to the case selection criteria. At this point, business experts who will carry out the later manual verification in order to evaluate precision and recall measures. If possible, the case studies involve two experts with two different expertise levels. First, the chief operating officer (COO) who is responsible for ensuring that business operations are efficient and effective and therefore knows the actual business processes carried out by his company. Secondly, the chief information officer (CIO) who is responsible for the information technology and computer systems supporting business operation.
2. Each LIS under study is implanted in an experimental environment: the source code is deployed, the database schema is built through the database scripts, the initial data is loaded, etc.
3. The tool that supports the proposal is used to obtain the different business process models from the legacy source code. The

Table 5
Legacy information systems under study.

Id	Name	Description	Architecture	Size (KLOC)
S1	AELG-Members (AELG, 2009)	Supports the administration of an organization of authors	Desktop application	23.5
S2	Tap CRM (Source Tap, 2009)	Is a sales force automation tool for sales management	Web application	49.6
S3	VillasanteLab (Villasante, 2010)	Manages a laboratory of the water and waste industry	Web application	28.8
S4	XuntaEadmin (Enxenio Inc, 2009)	Supports the electronic administration of a Spanish regional ministry	E-government system	320.2
S5	SIXA (SIXA, 2009)	Is a learning management system	Web application	140.6

execution hardware environment consists of a computer with a RAM memory of 4 GB and two processors of 2.67 GHz each.

4. The first sketch of the business processes obtained through the model transformation is improved by business experts.
   4.1 They fit the preliminary business processes with the reality of the organization, i.e., they can add tasks that should have been recovered but were not recovered, or remove tasks that were erroneously recovered.
   4.2 After business expert intervention, the accuracy of the business process models is evaluated by comparing each preliminary business process and its related business process that has been enhanced by business experts. We obtain the value of the proposed measures such as precision and recall by scoring the differences between the preliminary and enhanced business processes.
   4.3 Steps 2, 3 and 4 are repeated for each LIS under study.
5. The key information related to the generation of business processes (cf. step 3), along with the business expert intervention (cf. step 4), is collected for each model according to the data collection plan (cf. Section 4.4).
6. The data collected in the previous step is analyzed and interpreted to draw conclusions in order to answer the research questions. Section 4.5 shows the outgoing results obtained in this case study, while Section 4.6 provides a meta-analysis of the analysis carried out in order to obtain strengthened conclusions.

4.4. Data collection

Table 6 summarizes the most relevant data collected during execution of the case studies. This table shows: (i) the business process identifier; (ii) the study identifier; (iii) the business process name; (iv) the number of tasks recovered (before manual intervention); (v) the number of recovered relevant tasks, i.e., tasks marked as correct by experts; (vi) the number of recovered non-relevant tasks, i.e., tasks removed since they did not represent a business activity; (vii) the number of non-recovered relevant tasks, i.e., tasks added by experts; (viii) precision and (ix) recall values for each final business process; (x) the harmonic mean between them; (xi) the total number of tasks for each final business process; (xii) the transformation time in milliseconds; and finally (xiii) the time spent on manual post-intervention in seconds. Table 6 also shows the mean values for each study.

4.5. Analysis and interpretation

The collected data were analyzed to obtain the evidence chains from data to answer the research questions. In order to answer the question *AQ1*, Fig. 7 shows the box chart for precision and recall measures. The means of the distributions of the precision measure (0.51, 0.53, 0.66, 0.64, and 0.54) are always lower than the recall means (0.77, 0.83, 0.91, 0.65, and 0.66). Higher recall values signify that the proposed technique recovers very complete processes (i.e., it recovers most of the tasks from the current business processes). However, the lower precision values signify that the technique is imprecise (i.e., the number of recovered non-relevant tasks is very high) (see Fig. 7). The significant amount of non-relevant tasks is owing to the fact that some tasks are basically obtained from the source code and they are related to the technical nature, and do not therefore represent any piece of the embedded business knowledge.

The transformation time was also analyzed in order to answer *AQ2*. The means of the transformation time for the five systems were 24, 2143, 8589, 8931 and 2315 ms. These values were different because the average of the process size for each study was quite different: 52, 33, 107, 72 and 33 tasks (see Table 6). These time values seem feasible for the selected cases, since the size of

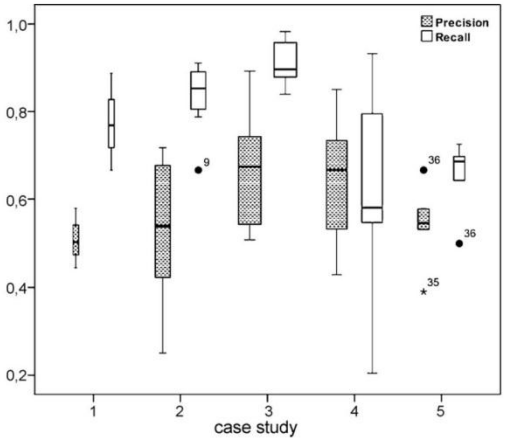


Fig. 7. Box chart of the precision and recall measures for each study.

the systems is above 20 *KLOC*. Nevertheless, the scalability of the procedure must be evaluated. Under the hypothesis that the time complexity of the procedure is theoretically linear (i.e., $O(n)$ considering 'n' as the number of tasks in a business process model) a linear regression model is established to check it and discover whether the proposal is therefore scalable.

The linear regression model considers the transformation time as a dependent variable and the size of the business processes as the independent variable. Fig. 8 shows the scatter charts of size/time for the five studies, along with their regression lines. Fig. 8 shows the regression line by grouping all values. The total regression line presents a positive linear relationship with $R^2 = 0.46$. The correlation coefficient R^2 (between −1 and 1) is the degree to which the real values of the dependent variable are close to the predicted values. The R^2 value obtained is a discrete result, but it is obtained as a consequence of lower R^2 values for individual studies like the first and fourth study, which have a few points to obtain interpolation regression lines with high statistical power level. In any case, the proposed linear regression model is suitable to explain the data obtained in this study, i.e., there is no quadratic or exponential relationship between the transformation time and the model size. The increase in time for larger systems will consequently be linear, and the time will thus be assumable. In conclusion, *AQ2* can therefore be answered as true, and the main research question *MQ* is also answered as true, i.e., the proposed business process recovery procedure makes it possible to obtain business process models from LISs in an effective and efficient manner.

Owing to the fact that the proposed technique is semi-automatic, the manual effort made by the business experts must be also analyzed, at least from a qualitative point of view. The selected business expert took an average of 362 s (6 min)for each business process model modified from the first sketch of the models retrieved by the tool. The manual time spent on this study is greater than the computational time. In fact, the manual intervention stage could be the bottleneck of the technique when it deals with large and complex system (see Fig. 9). Thereby, precision values around 55% signify that the retrieved models contains quite noise, which will imply a great effort during manual post-intervention. However, the manual time related to the proposed semi-automatic technique would be less than the time spent on redesigning a manual business process from scratch (Mutschler and Reichert, in press). The fault tolerance would be better with the proposed technique since

Table 6
Business process data collection for all case studies.

Subject ID	Study ID	Business process model name	# Rec tasks	# Rec rel tasks	# Rec non-rel tasks	# Non-rec rel tasks	Precision	Recall	*F*-measure	Size	Transf. time (ms)	Manual time (s)
1	S1	Category mgmt.	69	40	29	12	0.580	0.769	0.661	52	22	690
2	S1	Author mgmt.	141	71	70	9	0.504	0.888	0.643	80	35	520
3	S1	Reporting	36	16	20	8	0.444	0.667	0.533	24	14	240
		S1 mean	82	42	40	10	0.51	0.77	0.61	52	24	483
4	S2	Security mgmt.	69	30	39	4	0.435	0.882	0.583	34	1566	120
5	S2	Administration	45	26	19	7	0.578	0.788	0.667	33	630	300
6	S2	Chemical analysis mgmt.	80	51	29	5	0.638	0.911	0.750	56	3974	660
7	S2	Chemical calibration mgmt.	39	28	11	6	0.718	0.824	0.767	34	1184	540
8	S2	User mgmt.	64	16	48	3	0.250	0.842	0.386	19	782	180
9	S2	Chemical dilution mgmt.	39	16	23	8	0.410	0.667	0.508	24	1123	480
10	S2	Reporting	53	38	15	6	0.717	0.864	0.784	44	4981	120
11	S2	District mgmt.	36	18	18	2	0.500	0.900	0.643	20	2907	90
		S2 mean	53	28	25	5	0.53	0.83	0.64	33	2143	311
12	S3	Administration	134	69	65	10	0.515	0.873	0.648	79	4771	150
13	S3	Social-protection flat mgmt.	97	64	33	6	0.660	0.914	0.766	70	5895	280
14	S3	Document mgmt.	33	23	10	3	0.697	0.885	0.780	26	2724	540
15	S3	House rental mgmt.	290	221	69	9	0.762	0.961	0.850	230	23383	700
16	S3	Document renovation	204	169	35	3	0.828	0.983	0.899	172	10738	230
17	S3	House applicant mgmt.	213	190	23	9	0.892	0.955	0.922	199	13210	540
18	S3	Personal file mgmt.	134	74	60	9	0.552	0.892	0.682	83	5249	320
19	S3	Rural house mgmt.	118	63	55	12	0.534	0.840	0.653	75	10383	310
20	S3	Developer mgmt.	146	87	59	15	0.596	0.853	0.702	102	7034	280
21	S3	Renovation mgmt.	132	91	41	3	0.689	0.968	0.805	94	10086	430
22	S3	Emancipation grant mgmt.	122	62	60	7	0.508	0.899	0.649	69	5301	380
23	S3	Second-hand house mgmt.	105	76	29	9	0.724	0.894	0.800	85	4302	350
		S3 mean	144	99	45	8	0.66	0.91	0.76	107	8590	376
24	S4	Schedule mgmt.	104	73	31	16	0.702	0.820	0.756	89	7549	450
25	S4	Qualification mgmt.	73	56	17	5	0.767	0.918	0.836	61	14670	590
26	S4	Class group mgmt.	66	44	22	34	0.667	0.564	0.611	78	3218	550
27	S4	Evaluation	33	21	12	82	0.636	0.204	0.309	103	2862	430
28	S4	Remark mgmt.	87	74	13	22	0.851	0.771	0.809	96	17952	590
29	S4	Reporting	147	98	49	36	0.667	0.731	0.698	134	18289	230
30	S4	Notice mgmt.	45	25	20	18	0.556	0.581	0.568	43	1628	310
31	S4	Permission mgmt.	47	24	23	18	0.511	0.571	0.539	42	2411	290
32	S4	Student mgmt.	74	36	38	32	0.486	0.529	0.507	68	25261	460
33	S4	Subject mgmt.	21	9	12	8	0.429	0.529	0.474	17	862	520
34	S4	Teacher mgmt.	70	55	15	4	0.786	0.932	0.853	59	3547	400
		S4 mean	70	47	23	25	0.64	0.65	0.63	72	8932	438
35	S5	Provider Mgmt.	59	23	36	10	0.390	0.697	0.500	33	1905	130
36	S5	Product mgmt.	6	4	2	4	0.667	0.500	0.571	8	172	240
37	S5	Event mgmt.	47	25	22	12	0.532	0.676	0.595	37	1252	180
38	S5	Sending mgmt.	64	37	27	14	0.578	0.725	0.643	51	1674	240
39	S5	Reporting	33	18	15	10	0.545	0.643	0.590	28	7471	150
40	S5	Relationship mgmt.	55	30	25	13	0.545	0.698	0.612	43	1419	270
		S5 mean	44	23	21	11	0.54	0.66	0.59	33	2316	202
		Total mean	85.8	54.8	31.0	12.6	0.601	0.775	0.664	67.4	5810.9	362
		Standard deviation	58.2	47.3	18.0	13.9	0.138	0.166	0.138	48.3	6430.0	172

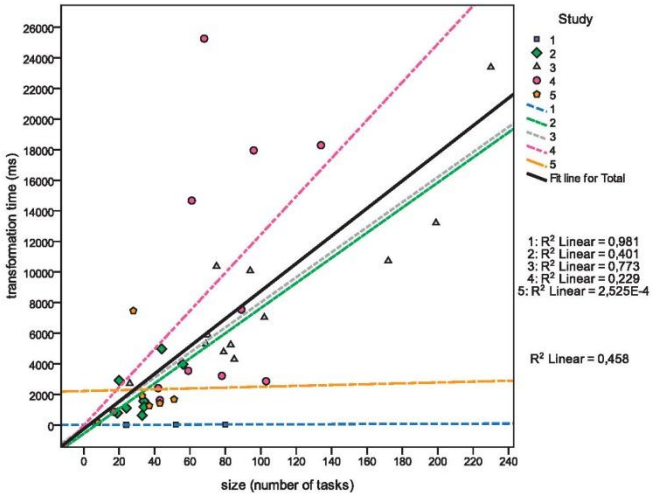


Fig. 8. The size/time scatter chart considering a linear regression model.

it provides a first sketch of the actual business processes. Furthermore, redesign from scratch by manual solutions has at least two drawbacks. Firstly, manual solutions do not consider the business knowledge that is embedded in LISs, and which is not present in any other artifact. Secondly, business process redesign copes with the numerous technological, organizational and project-driven cost factors related to the context of BPM projects. For example, manual redesign can be influenced by intangible impact factors like available process knowledge or end user fears (Mutschler and Reichert, in press).

The manual intervention time, which is high, could be reduced by adding methods to aid decision-making of business experts. For example, the tool could analyze the business processes and provides some suggestions to improve the obtained models. Particularly, such analysis could determine similar fragments in two different business processes, thus it may suggest joining those processes. Also, the analysis could evaluate metrics such as the number of tasks and the intricacy degree (ratio of sequence flows regarding tasks) and suggests splitting a process into two or more processes according to the subsets of data that some process fragments manage. As a result, the tool will provide a prioritized list of candidate actions, for which business experts could select the most suitable actions and/or provides additional modifications.

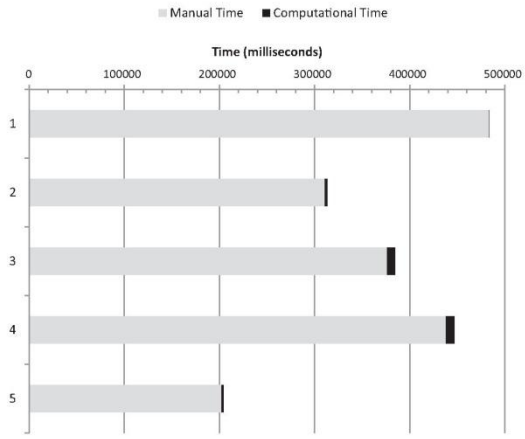


Fig. 9. Relationship between manual and computational time in the five case studies.

4.6. Meta-analysis study

Several statistical methods (e.g., meta-analysis, significance level combination or vote counting) allow researchers to accumulate and interpret a set of results obtained through different empirical studies that are inter-related because they check similar hypotheses (Hedges and Olkin, 1985; Wolf, 1986). In the study presented here, we have used meta-analysis to strengthen our conclusions.

While simple analysis attempts to evaluate a sole study in isolation, meta-analysis combines results of several ones. Meta-analysis is a set of statistical techniques that combine the different effect size measures or treatment effect of various individual studies. There are several metrics to obtain this value, e.g. the means difference and the correlation coefficients, among others (Hedges and Olkin, 1985). The objective is to obtain a global treatment effect of all the cases studies.

As effect size measures may come from different environments and heterogeneous studies or experiments, Meta-analysis first obtains standardized measures of each one. After that, the global effect size is obtained as a weighted average of standardized measures, in which the most commonly used weights are the sample size or the standard deviation. Finally, together with the estimation of the global effect size, meta-analysis provides an estimated confidence interval and a p-value which is used to decide on the meta-analysis hypotheses. Most meta-analysis techniques are automated by tools. The meta-analysis presented in this work has been obtained by using the *Meta-Analysis v2* tool (Biostat Inc Comprehensive Meta-Analysis v2, 2006).

The objective of the meta-analysis presented here is the evaluation of the difference in means for the recall and precision measures obtained in each case study. Once the overall effect size is

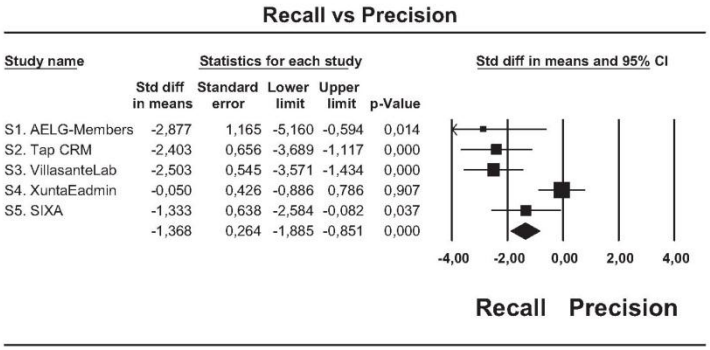


Fig. 10. Meta-analysis results and forest plot (recall vs. precision).

calculated, we can provide a confidence interval or a *p*-value which allows us to decide on the meta-analysis hypotheses, such as those which can be found in other empirical software engineering works (Cruz-Lemus et al., 2009; Dybå et al., 2007). Our meta-analysis hypothesis can be stated as: *recall is as accurate as precision in our technique for recovering business processes (H_0).*

Fig. 10 shows the forest plot after applying the meta-analysis. The left-side of the plot lists the names of the studies. The right-side of the plot shows the measure of effect, i.e., difference of means, for each of these studies (represented by a square) incorporating confidence intervals (represented by horizontal lines). The area of each square is proportional to the study's weight in the meta-analysis. Finally, the overall meta-analyzed measure of effect is represented as a diamond.

The results obtained (see Fig. 10) allow us to reject this hypothesis, as they indicate that recall works more efficiently than precision. These results agree with those stated in the previous sub-section and they are statistically significant at a level of 0.05, so we can conclude that the idea of recall being more accurate than precision in our business technique is statistically confirmed.

The result obtained for recall and precision measures is usual since there is an inverse relationship between both measures (see Fig. 11). The precision value should, ideally, always be 1 for any recall value, but this is not possible in practice. In fact, the proposed method could increase its Precision value by recovering fewer tasks since the precision measure has the total number of recovered task in the denominator. To maintain the higher Recall value, the reduction should only focus on non-relevant tasks to achieve a higher Precision while Recall is as higher as possible. It is however a hard task and some relevant tasks will probably be also discarded. As a result, while Recall has been reduced a little bit, Precision has been increased much more regarding the reduction of Recall. This hypothetical result would be more desirable than the obtained result, since the Precision and Recall values would be more balanced.

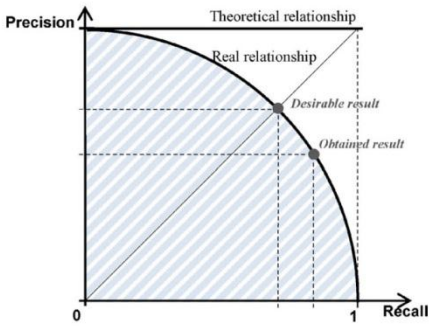


Fig. 11. Relationship between precision and recall measures and obtained/desirable results.

In order to jointly evaluate both measures, the *F*-measure values, which were 0.61, 0.64, 0.76, 0.63 and 0.59 for the five studies (see Table 6), were also analyzed. These values were compared with a reference value of around 0.55, which is considered in other model recovery experiences (Lucrédio et al., 2008). The values obtained for all studies were clearly above 55%. Moreover, the total means obtained by grouping all studies were precision = 0.60, recall = 0.77 and *F*-measure = 0.66 (see Table 6), which were above the reference value. As a result, *AQ1* can be positively answered, i.e., the proposed technique recovers business processes from LISs with adequate accuracy levels. However, the *F*-measure value might be slightly improved by obtaining more balanced precision and recall values.

4.7. Evaluation of validity

This stage evaluates whether the results are true and not biased for the whole population to which we wish to generalize the results. This section thus shows the threats to the validity of the family of case studies.

According to the *internal validity*, an average population of 40 business process models was recovered, and it is thus possible to obtain statistically representative results. Nevertheless, the study may be replicated by using more systems to attain a larger population. In addition, there are two determining factors related to obtaining the results presented here: (i) the supporting tool used to obtain the business processes could be a factor that affects the transformation time values, and (ii) if the study is replicated with other cases involving different business experts, the measure values might have some deviations, since each expert provides his/her subjective viewpoint. The replication of the study using different configurations is a possible means to mitigate these threats.

Moreover, according to the *construct validity*, the selected measures were adequate to answer the research questions in an appropriate manner. Both precision and recall were reused from the information retrieval field, in which these metrics have an adequate maturity level. However, the way in which such measures are evaluated could be a threat, since these consider tasks as the sole scoring element. To mitigate this threat, alternative mechanisms could be considered such as the combination of various scoring elements at the same time (e.g., tasks and sequence flows).

Anyway, tasks are, or are not, considered as relevant tasks by assessing their accuracy as well as their related sequence flows, data objects, and so on.

Finally, *external validity* is concerned with the generalization of the results. This study considers, as the whole population, traditional LISs following the object-oriented paradigm. The results obtained could therefore be strictly generalized to this population. However, the specific platform of the selected case is a threat that should be noted. This threat could be mitigated by replicating the study using systems that follow different platforms (not solely object-oriented).

5. Conclusions and future work

This paper has addressed the problem of business process recovery from legacy information systems in order to preserve valuable business knowledge. The preservation of this knowledge facilitates the alignment between the actual business behavior in an organization and their information systems. In addition, business knowledge preservation allows maintainers to modernize legacy information systems, extending their lifecycles and therefore improving the ROI. This paper provides a technique with which to recover business process, and which is framed in MARBLE, an ADM-based framework. The technique is characterized by three main features: (i) it focuses on legacy object-oriented code as the main source of knowledge, (ii) it uses static analysis as a reverse engineering technique to extract the information needed; and (iii) it follows the model-driven development principles, i.e., it considers different models at different abstraction levels and a set of model transformations between them. A supporting tool is additionally provided in order to automate the technique and facilitate its adoption.

In order to validate the proposal, this paper presents a family of industrial case studies, which were carried out over 2 years with five real-life legacy information systems. The case studies evaluate the effectiveness and efficiency of the technique. The effectiveness is evaluated using precision and recall measures. These metrics are appropriate to discover the accuracy (precision) and completeness (recall) of each of the business processes recovered. Moreover, in order to evaluate the efficiency, the study evaluates the recovery time with regard to the size of each business process model. It therefore checks the scalability of the technique to large legacy information systems.

The results obtained from the study were also meta-analyzed to obtain strengthened conclusions. The result obtained shows that the technique is suitable to recover business processes in an effective and efficient manner. However, according to the effectiveness, the recall values were better than the precision values. We believe that these results were obtained because much of the work was, in several cases, basically recovered from technical code.

Our future work will address the threats to validity identified. The study will be replicated with more legacy information systems based on other platforms or languages in order to compare and generalize (if possible) the results obtained. Besides improving the empirical validation, the technique will also be improved by incorporating the dynamic analysis in order to extract more and valuable business knowledge during system execution. The final objective is to obtain more balanced values of recall and precision, although the *F*-measure values must be simultaneously preserved.

Acknowledgement

This work was supported by the Spanish FPU Programme, and by the R&D projects ALTAMIRA (JCCM, PII2I09-0106-2463), MOTERO (JCCM and FEDER, PEII11-0366-9449), MEDUSAS (CDTI (MICINN), IDI-20090557) and PEGASO/MAGO (MICINN and FEDER, TIN2009-13718-C02-01).

References

Alarcos Research Group MARBLE Tool, 2011. Available from: http://www.business-processarcheology.org/downloads/MARBLE.rar [cited 10.06.11].
Biostat Inc Comprehensive Meta-Analysis v2, 2006, available from: http://www.meta-analysis.com/ (cited 17.08.10).
Cai, Z., Yang, X., Wang, W.,2009. Business process recovery for system maintenance—an empirical approach. In: 25th International Conference on Software Maintenance (ICSM'09). IEEE Computer Society, Edmonton, Alberta, Canada, pp. 399–402.
Canfora, G., Di Penta, M., Cerulo, L., 2011. Achievements and challenges in software reverse engineering. Communications in ACM 54 (4), 142–151.
Chang, J.F., 2006. Business Process Management Systems: Strategy and Implementation. Auerbach Publications.
Chikofsky, E.J., Cross, J.H., 1990. Reverse engineering and design recovery: a taxonomy. IEEE Software 7 (1), 13–17.
Cornelissen, B., et al., 2009. A systematic survey of program comprehension through dynamic analysis. IEEE Transactions on Software Engineering 35 (5), 684–702.
Cruz-Lemus, J.A., et al., 2009. Assessing the understandability of UML statechart diagrams with composite states—a family of empirical studies. Empirical Software Engineering 14, 685–719.
Di Francescomarino, C., Marchetto, A., Tonella, P.,2009. Reverse engineering of business processes exposed as web applications. In: 13th European Conference on Software Maintenance and Reengineering (CSMR'09). IEEE Computer Society, Fraunhofer IESE, Kaiserslautern, Germany, pp. 139–148.
do Nascimento, G.S., et al.,2009. A method for rewriting legacy systems using business process management technology. In: 11th International Conference on Enterprise Information Systems (ICEIS'09). INSTICC, Milan, Italy, pp. 57–62.
Dybå, T., et al., 2007. Are two heads better than one? on the effectiveness of pair programming. IEEE Software 24 (6), 10–13.
Eisenbarth, T., Koschke, R., Simon, D., 2003. Locating features in source code. IEEE Trans. Softw. Eng. 29 (3), 210–224.
Enxenio Inc Enxenio web site, 2009. http://www.enxenio.es (cited 01.03.09).
Enxenio Inc. AELG-Members, 2009. http://www.aelg.org/ (cited 18.08.2011).
Enxenio Inc SIXA, 2009. http://www.sixa.es/es (cited 18.08.11).
Ghose, A., Koliadis, G., Chueng, A., 2007. Process Discovery from Model and Text Artefacts. In: IEEE Congress on Services (Services'07), Salt Lake City, UT.
Günther, C.W., van der Aalst, W.M.P., 2007. A generic import framework for process event logs. In: Business Process Intelligence Workshop (BPI'06), LNCS 4103, pp. 81–92.
Hedges, L.V., Olkin, I., 1985. Statistical Methods for Meta-analysis. Academia Press.
Heuvel, W.-J.v.d., 2006. Aligning Modern Business Processes and Legacy Systems: A Component-Based Perspective (Cooperative Information Systems). The MIT Press.
Ingvaldsen, J.E., Gulla, J.A., 2008. Preprocessing Support for Large Scale Process Mining of SAP Transactions. In: Business Process Intelligence Workshop (BPI'07), LNCS 4928, pp. 30–41.
ISO/IEC, ISO/IEC DIS 19506, 2009. Knowledge Discovery Meta-model (KDM), v1.1 (Architecture-Driven Modernization). ISO/IEC, p. 302, http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2=080&ics3=&csnumber=32625.
Khusidman, V., Ulrich, W., 2007. Architecture-Driven Modernization: Transforming the Enterprise. DRAFT V.5. OMG, p. 7, http://www.omg.org/docs/admtf/07-12-01.pdf.
Koskinen, J., et al., 2004. Estimation of the Business Value of Software Modernizations. Information Technology Research Institute, University of Jyväskylä.
Lehman, M.M., 1984. Program evolution. Information Processing & Management 20 (1–2), 19–36.
Lewis, G.A., Smith, D.B., Kontogiannis, K., 2010. A Research Agenda for Service-Oriented Architecture (SOA): Maintenance and Evolution of Service-Oriented Systems. Software Engineering Institute, p. 40.
Lucrédio, D., Fortes, R.P.M., Whittle, J., 2008. MOOGLE: A Model Search Engine, in 11th international conference on Model Driven Engineering Languages and Systems. Springer-Verlag, Toulouse, France, pp. 296–310.
Milner, R., Parrow, J., Walker, D., 1992. A calculus of mobile processes. I. Inf. Comput. 100 (1), 1–40.
Mutschler, B., Reichert, M., 2012. Understanding the costs of business process management technology. Advances in Business Process Management, http://dbis.eprints.uni-ulm.de/742/. Access Date: 21/12/2011.
Newcomb, P.,2005. Architecture-driven modernization (ADM). In: Proceedings of the 12th Working Conference on Reverse Engineering. IEEE Computer Society.
OMG, 2009a. Architecture-Driven Modernization Standards Roadmap. February 2009; Available from: http://adm.omg.org/ADMTF%20Roadmap.pdf (cited 29.10.09).
OMG, 2009b. Business Process Model and Notation (BPMN) 2.0. Object Management Group, p. 496.
Paradauskas, B., Laurikaitis, A., 2006. Business Knowledge Extraction from Legacy Informations Systems. Journal of Information Technology and Control 35 (3), 214–221.
Paradauskas, B., Laurikaitis, A., 2011. Extracting Conceptual Data Specifications from Legacy Information Systems. Elektronika ir elektrotechnika 107 (1), 46–50.

Pérez-Castillo, R. KDM to BPMN Transformation implemented in QVT Relations. 2011 26/04/2011, available from: http://alarcos.esi.uclm.es/per/rpdelcastillo/modeltransformations/KDM2BPMN.htm (cited 02.05.2011).
Pérez-Castillo, R., et al.,2009a. MARBLE: A Modernization Approach for Recovering Business Processes from Legacy Systems. In: International Workshop on Reverse Engineering Models from Software Artifacts (REM'09). Simula Research Laboratory Reports, Lille, France, pp. 17–20.
Pérez-Castillo, R., et al., 2009b. PRECISO: A Reengineering Process and a Tool for Database Modernisation through Web Services. In: 24th Annual ACM Symposium on Applied Computing (SAC'09), Hawaii, USA, pp. 2126–2133.
Pérez-Castillo, R., et al.,2010a. Business Process Patterns for Software Archeology. In: 25th Annual ACM Symposium on Applied Computing (SAC'10). ACM, Sierre, Switzerland, pp. 165–166.
Pérez-Castillo, R., García-Rodríguez de Guzmán, I., Piattini, M.,2010b. Implementing Business Process Recovery Patterns through QVT Transformations. In: International Conference on Model Transformation (ICMT'10). Springer-Verlag, Málaga, Spain, pp. 168–183.
Pérez-Castillo, R., García Rodríguez de Guzmán, I., Piattini, M., 2011. Knowledge discovery metamodel—ISO/IEC 19506: a standard to modernize legacy systems. Computer Standards & Interfaces Journal 33 (6), 519–532.
Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14 (2), 131–164.
Sneed, H.M.,2005. Estimating the Costs of a Reengineering Project. In: Proceedings of the 12th Working Conference on Reverse Engineering. IEEE Computer Society, pp. 111–119.
Source Tap, Source Tap CRM, 2009. http://sourcetapcrm.sourceforge.net/.
van der Aalst, W., Reijers, H., Weijters, A., 2007. Business process mining: an industrial application. Information Systems 32 (5), 713–732.
Villasante, J.M., 2010. Laboratorio de Análisis J.M. Villasante. http://www.laboratoriovillasante.com/index.php [cited 18.08.11].
Visaggio, G., 2001. Ageing of a data-intensive legacy system: symptoms and remedies. Journal of Software Maintenance 13 (5), 281–308.
Wang, X., et al.,2004. Business rules extraction from large legacy systems. In: Proceedings of the Eighth Euromicro Working Conference on Software Maintenance and Reengineering (CSMR'04). IEEE Computer Society.
Weerakkody, V., Currie, W.,2003. Integrating business process reengineering with information systems development: issues and implications. In: Proceedings of the 2003 international conference on Business process management. Springer-Verlag, Eindhoven, The Netherlands, pp. 302–320.
Weske, M., 2007. Business Process Management: Concepts, Languages, Architectures. Springer-Verlag, Leipzig, Berlin Heidelberg, Germany, p. 368.
Wolf, F.M., 1986. Meta-analysis: Quantitative Methods for Research Synthesis. Sage Publications.
Zou, Y., et al.,2004. Model-driven business process recovery. In: Proceedings of the 11th Working Conference on Reverse Engineering (WCRE 2004). IEEE Computer Society, pp. 224–233.

Ricardo Pérez-Castillo holds the MSc degree in Computer Science from the University of Castilla-La Mancha, and he is currently a PhD student in Computer Science. He Works at the Instituto de Tecnologías y Sistemas de Información (ITSI) at the University of Castilla-La Mancha. His research intererests include architecture-driven modernization, model-driven development and business process mining. Contact him at Escuela Superior de Informática, Paseo de la Universidad 4, 13071-Ciudad Real, Spain; Ricardo.PdelCastillo@uclm.es.

Ismael Caballero is assistant professor at the University of Castilla-La Mancha and belongs to the Alarcos Research Group at the UCLM. He holds the PhD degree in Computer Science from the University of Castilla-La Mancha. His research interests include software and data quality, database design and software development based on quality. Contact him at Escuela Superior de Informática, Paseo de la Universidad 4, 13071-Ciudad Real, Spain; Ismael.Caballero@uclm.es.

Ignacio García-Rodriguez de Guzmán is assistant professor at the University of Castilla-La Mancha and belongs to the Alarcos Research Group at the UCLM. He holds the PhD degree in Computer Science from the University of Castilla-La Mancha. His research interests include software maintenance, software modernization and service-oriented architecture. Contact him at Escuela Superior de Informática, Paseo de la Universidad 4, 13071-Ciudad Real, Spain; Ignacio.GRodriguez@uclm.es.

Mario Piattini is full professor at the UCLM. His research interests include software quality, metrics and maintenance. He holds the PhD degree in Computer Science from the Technical University of Madrid, and leads the Alarcos Research Group at the Universidad de Castilla-La Mancha. He is CISA, CISM e CGEIT by ISACA. Contact him at Escuela Superior de Informática, Paseo de la Universidad 4, 13071-Ciudad Real, Spain; Mario.Piattini@uclm.es.

CHAPTER 6

Conclusions

> *The research proposal facilitates the extraction of business rules (through business process discovering) that must be incorporated into new versions of information systems* […] *to extend the lifespan of legacy information systems and, therefore, it also help to improve the ROI.*

6. CONCLUSIONS

This chapter discusses the conclusions of this PhD Thesis, and is organized as follows: Section 6.1 provides a brief summary of the Thesis. Section 6.2 then analyzes the initial research hypothesis and the attainment of the research goals. Section 6.3 summarizes the main contributions of the Thesis and its implication for academia and industry. Section 6.4 states future research lines. Finally, Section 6.5 shows the whole set of papers published throughout the PhD Thesis.

6.1. Summary

Business processes are a valuable asset for many companies since they provide them with a means to comprehend and carry out their daily operations. Companies that appropriately manage their business processes have advantages over other companies in terms of competitiveness. Business process management allows companies to obtain a better understanding of their intrinsic operations, thus allowing them to enhance and adapt these operations according to new business goals or changes in the environment. Moreover, business process activities are often automated by enterprise information systems. As a result, business process management also facilitates the alignment between business processes and enterprise information systems.

Despite the importance of business processes and their management, many companies and organizations ignore this issue and have no available business process representations. When these companies decide to represent and manage their business processes, they must deal with the discovery of the implicit business processes that they carry out every day. One of the most important sources of knowledge with which to discover business processes is legacy information systems. Legacy information systems embed much business knowledge which is often not available anywhere else as a consequence of the uncontrolled maintenance of legacy information systems over time. An alternative source of knowledge is the organization's business experts who can carry out a green-field redesign of business processes. This manual option is, however, time consuming and error-prone in comparison to semi-automatic business process discovery from information systems. What is more, much of the business knowledge that is embedded in legacy information systems might be ignored by manual redesign from scratch.

This PhD Thesis addresses the challenge of discovering and reconstituting business processes from legacy information systems in a semi-automatic manner. It particularly focuses on two scenarios. The first scenario occurs when companies that have never explicitly represented their business processes wish to adopt business process management. In this scenario, the action consists of taking the existing information systems and obtaining business processes from them. The second scenario occurs in companies or organizations that have

already represented and managed their business processes. However, legacy information systems evolve together with the implicit business process activities supported by these systems. Unfortunately, business process representations do not evolve together with legacy information systems and are misaligned with them. In this scenario, the goal is to discover and reconstitute actual business processes in line with information systems. This scenario allows companies to modernize their legacy information systems by taking into account the actual business processes that they are currently carrying out.

In order to contribute to the solution of the aforementioned challenge, this Thesis proposes MARBLE, a framework with which to discover and retrieve business processes from legacy information systems. MARBLE follows the specifications and standards proposed by the ADM initiative. This initiative advocates carrying out traditional reengineering by considering model-driven development principles. There are basically two principles: (i) the homogeneous treatment of all the artifacts involved as models, representing them at different abstraction levels, and (ii) the automation of model transformations between these abstraction levels. These principles help to solve the automation and formalization problems of traditional reengineering. MARBLE is a generic framework based on ADM which focuses on the reverse engineering stage to obtain business processes form different software legacy artifacts.

The *generic framework* signifies that MARBLE is able to define the guidelines with which to develop the specific techniques of business process discovery. The key point that allows MARBLE to be such a generic framework is the establishment of four abstraction levels (i.e., four kinds of models) and three model transformations between them. The four abstraction levels are: L0, which defines the actual software artifacts of a legacy information system (e.g., source code or database); L1, which represents the PSM models of each of the software artifacts considered (e.g., a code model, a database model); L2, which combines all the models retrieved in a KDM repository representing a PIM model; and finally L3, which defines business process models according to the BPMN specification. MARBLE-framed techniques must specifically define the three model transformations between these levels.

This PhD Thesis presents two concrete techniques with which to discover business processes framed in MARBLE. The first technique employs, as a reverse engineering technique, static analysis, while the second uses dynamic analysis which takes advantage of runtime information. The static technique syntactically analyzes source code and then recognizes a set of patterns in KDM models in order to obtain a first sketch of business process models. The set of business patterns can be extended with additional patterns. Business experts then analyze and refine the first sketch to fit business process models to the actual business activity of the organization. This technique is therefore considered to be a business process archeology technique.

While the static technique implies manual intervention at the end of the process, the dynamic technique first considers business expert intervention in order to instrument source code. The instrumented version of a legacy information system is able to record events during

system execution. These events represent the execution of business activities that are supported by different parts of source code. Event logs are then analyzed in order to mine business process models. This technique is therefore considered to be a process mining technique rather than a business process archeology technique.

Both the static and the dynamic technique are supported by tools especially developed during this Thesis. These tools support the whole framework and facilitate the demonstration of the feasibility of the proposed techniques throughout a set of industrial case studies. These case studies were rigorously planned and conducted by following a formal protocol especially developed for software engineering case studies. These studies were conducted with a great variety of cases, e.g., an author management system, an open source CRM system, an enterprise information system of the water and waste industry, an e-government system used in a Spanish local e-administration, a high school learning management system, and a healthcare evaluation system used in Austrian hospitals. While the static technique proved to be more exhaustive than the dynamic technique, the dynamic technique proved to be more accurate as regards the number of business activities appropriately discovered. The results obtained from all these case studies demonstrate that both the static and the dynamic technique are suitable to obtain business processes form real-life legacy information systems. Furthermore, the combination of both techniques proved to be more accurate and complete than either technique in isolation.

6.2. Research Hypothesis and Analysis of Goals

The objective of this section is to evaluate the achievement of the research goals established at the beginning of this Thesis. The main research hypothesis of this PhD Thesis was presented in Chapter 1.

> ***It is feasible to construct a framework by using the ADM approach to retrieve business processes embedded in different kinds of legacy information systems in a standardized and automatic manner.***

In order to evaluate the research hypothesis, this Thesis established the main research objective:

> ***To define an ADM-based framework with which to discover and reconstitute business processes from legacy information systems.***

A set of partial objectives which had to be addressed in order to achieve the main research objective was also stated. A description of the achievement of each of these partial objectives is presented hereafter:

O1. *To carry out an in-depth study of the business process mining and archeology field in the context of software modernization which will provide the state-of-the-art of this Thesis.*

This objective has been fulfilled by carrying out a systematic literature review, which was the knowledge base for the publications presented in Chapter 2.

O2. *To define business process archeology techniques with which to extract embedded business knowledge form different software artifacts (both static and dynamic techniques).*

This objective has been achieved by proposing the static and dynamic techniques framed in MARBLE. In addition, MARBLE defines the guidelines needed to consider additional software artifacts and reverse engineering techniques in future techniques (see Chapter 3).

O3. T*o establish a technique with which to generate event logs from traditional, legacy information systems in order to reuse existing business process mining techniques.*

This objective has been addressed by the dynamic techniques, which partially obtain standardized event logs that can be used as the input for a wide variety of process mining techniques and algorithms in order to discover business processes (see Section 3.3).

O4. *To represent and integrate all the business knowledge analyzed in a common and platform-independent model repository to be used to retrieve business process models.*

Both MARBLE and the particular techniques have been defined by using the KDM standard as the mechanism to represent independent-technology repositories of knowledge (see Section 3.1).

O5. *To establish model transformation between the common knowledge repository and the target business processes and define traceability methods between elements transformed at different abstraction levels in order to facilitate feature location in business process models.*

This objective has been fulfilled since KDM repositories populated by the proposed techniques are used homogeneously through the discovery of business processes models by applying a set of business process patterns. The proposed transformations also introduce traceability information in the business processes generated, which is used to locate features (see Chapter 3).

O6. *To propose a technique with which to discover cross-cutting business process views according to different aspects or features of the legacy information systems.*

This objective has been addressed by the combination of static and dynamic techniques to provide partial views of business processes. Each view represents a fragment of the business processes that are obtained as regards a particular piece of source code.

O7. *To develop software tools to support and semi-automate the techniques proposed.*

Both the static and the dynamic techniques have been fully supported by tools especially developed in this Thesis. Chapter 4 particularly shows a supporting tool for the static technique, which has been used in several real projects.

O8. *To conduct an empirical validation of all the techniques used through the supporting tools by applying the techniques to real-life information systems in industrial case studies.*

This objective has been attained by carrying out seven case studies with real-life information systems. The supporting tools assisted in the realization of all these studies.

O9. *To carry out a meta-analysis of the combined results obtained from all the case studies in order to establish measures and indicators with which to assess the proposed techniques in comparison with similar existing ones.*

The isolated results obtained from these case studies have also been analyzed in combination (cf. Chapter 5). Meta-analysis techniques and the Bender method have been used to aggregate the individual results.

Based on the attainment and fulfillment of the partial objectives, the main objective of the Thesis has been fulfilled. An ADM-based framework has been proposed to retrieve business processes embedded in different kinds of legacy information systems. The MARBLE framework has additionally been defined in a standardized and semi-automatic manner. In conclusion, the main research hypothesis can be answered positively.

6.3. Main Contributions and Implications

The main contribution of this PhD Thesis is MARBLE, a generic ADM-based framework with which to obtain business processes from legacy information systems. The main implication of this contribution is that MARBLE is generic and based on international standards, and is not therefore limited to only one concrete technique or platform. This is the key difference with regard to related works in literature, which often present *ad hoc* or/and non-standard solutions. The use of MARBLE as a generic framework allows concrete techniques to be instantiated by following the standards and principles of MARBLE. This fact implies the future extension of MARBLE by proposing new techniques.

Both the static and the dynamic techniques demonstrate the possibility of defining concrete techniques from MARBLE. Unlike most proposals in literature, both techniques have been successfully applied to various real-life legacy information systems, thus demonstrating its applicability in real environments. In fact, the conduction of different industrial case studies is another important contribution since these studies facilitates the transference of technology to industry.

The conduction of different case studies has provided some evidence with which to compare the static and the dynamic techniques. The static and dynamic approaches are the two

main choices with which to discover the current business processes. The main difference between both approaches is whether or not the technique considers the system execution information. Despite the solutions proposed in literature, there are no previous empirical studies showing evidence of which approach is better, and under what conditions. Case studies have employed recall and precision measures to evaluate how techniques obtain business processes. Recall and precision respectively mean the completeness and accuracy of the business process models obtained. Case studies show that the completeness of business processes obtained with the static technique (around 75%) is higher than the completeness obtained with the dynamic technique (50% on average). However, the dynamic solution provides a higher accuracy (around 80%) than the static technique (60% on average). With regard to the efficiency of both techniques, the static technique is better since the dynamic technique often needs to instrument existing source code, which implies additional manual intervention.

The result of the comparison between the static and dynamic techniques has certain implications. Dynamic analysis is more powerful and accurate than static analysis. Nevertheless, the applicability of dynamic analysis has some limitations in real organizations since it frequently requires source code modifications in order to aggregate traces to collect system execution information. In practice, source code modifications are not always possible since legacy information systems may be in the production stage. This is a controversial point related to the usage of the dynamic technique in companies that have systems that manage confidential and sensitive information (e.g., bank, healthcare or e-government systems). Many of these organizations might not accept the usage of an automatically modified version of their information systems. The static technique can therefore be used in companies to ensure that the discovery of business processes is carried out without affecting the behavior of the original information system, i.e. in a non-invasive manner.

Moreover, there exist implications regarding all the possible usages of business processes obtained by applying the MARBLE-based techniques. As was introduced in Chapter 1, business processes discovered from legacy information systems can be used with two main objectives. Firstly, the business processes retrieved represent the actual, current processes carried out by an organization. These business processes may be used to discover (if it is the first discovery) or align business processes (when a previous version exists) with the actual business activities of a company. Unfortunately, there are manual business activities that might not be retrieved. This is why this Thesis proposes semi-automatic techniques to obtain business processes by focusing on legacy information systems. Nevertheless, in comparison to other approaches that follow manual redesign from scratch, MARBLE-based techniques inspect business knowledge embedded in legacy information systems, which may ignore manual business activities.

The second usage of business processes consists of the software modernization of legacy information systems. When a legacy information system is modernized, evolved or migrated, the whole behavior must be preserved in the new version. The research proposal facilitates the extraction of business rules (through business process discovery) which must be incorporated into new versions of information systems. A further implication of this Thesis is consequently

its contribution towards extending the lifespan of legacy information systems, and the fact that it therefore also helps to improve the ROI.

6.4. Future Research Lines

The future research lines related to this PhD Thesis are:

- **To improve the precision of the business process models retrieved.** The first sketch of business processes obtained by reverse engineering is often fine-grained and very intricate. The development of clustering techniques will therefore help to obtain more accurate, understandable and less complex business processes. Thanks to this automatic refinement, the effort related to manual post-intervention by business experts will be minimized or even avoided, and the precision of business process models will in turn be increased.
- **To refactorize the business process models retrieved.** Refactoring techniques can be employed to obtain alternative business processes, thus reducing the complexity of the business process models retrieved while preserving the external behaviour. Refactoring techniques allow more precise and understandable business process models to be obtained.
- **To incorporate alternative business knowledge sources.** MARBLE is a large framework which advocates considering different legacy software artifacts. Indeed, KDM facilitates the representation of knowledge extracted from different artifacts at different abstraction levels. This research line will provide alternative techniques framed in MARBLE which will consider, for example, legacy relational databases or user interfaces. On the one hand, databases can provide all the data objects used by business activities. Additionally, business process models can be mapped with the legacy data model, which is a valuable insight to align business processes with modernized information systems. On the other hand, user interfaces together with dynamic analysis is a powerful mechanism by which to discover the real sequence of operations carried out by users who actually interact with the legacy information system.
- **To strengthen empirical validation.** In order to mitigate the aforementioned threats to the validity of the case studies, a third research line will focus on case study replication with different configurations and different cases. The case studies will be conducted with different legacy information systems (e.g., different programming languages, technologies, size, complexity, etc.); by applying different process mining tools and algorithms to the event logs retrieved; and so forth.
- **To develop and maintain supporting tools.** In line with these future research lines, we have planned to develop and maintain supporting tools. Since MARBLE Tool has been released as an *Eclipse*™ plug-in, it can be easily extended and integrated into other

related tools, e.g., tools supporting both future MARBLE-based techniques and other related modernization and mining tools.

6.5. Results

Despite the fact that this PhD Thesis is presented as a collection of ten papers, the results of the research performed in this PhD Thesis have been published in others papers in diverse journals, conferences and books related to Software Engineering. Table 2 shows all the research works that have been published during the PhD Thesis in addition to other submitted publications under review.

Table 2. Summary of research works published during the PhD Thesis

	Published	Submitted	Total
Indexed International Journals	8	4	12
International Conferences (CORE A)	2		2
International Conferences (CORE B)	12		12
Other International Conferences	7	1	8
International Book Chapters	2		2
Latin American Journals	3		3
Latin American Conferences	8		8
Latin American Book Chapters		1	1
TOTAL	**42**	**6**	**48**

6.5.1. Published Papers

This section presents the research works that have already been published or accepted for publication. It divides the publications into various categories: JCR[1], indexed international journals; ITA, international *CORE*[2] *A* conferences; ITB, International *CORE B* conferences; ITC, other international conferences and workshops; CHP, book chapters; LAJ, Latin-American journals; and finally, LAC, Latin-American conferences.

Indexed International Journals

JCR-1.Pérez-Castillo, R., I.G.-R.deGuzmán, and M. Piattini, Business Process Archeology using MARBLE. Information and Software Technology, 2011. 53: p. 1023–1044. Impact factor: 1.507.

[1]http://ip-science.thomsonreuters.com/
[2]http://www.core.edu.au/

JCR-2.Pérez-Castillo, R., I. García Rodríguez de Guzmán, M. Piattini, and Á.S. Places, A Case Study on Business Process Recovery using an E-Government System. Software Practice & Experience Journal, 2011: p. 42(2): p. 159–189. Impact factor: 0.573.

JCR-3.Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, and M. Piattini, Process Mining through Dynamic Analysis for Modernizing Legacy Systems. IET Software Journal, 2011. 5(3): p. 304–319. Impact factor: 0.671.

JCR-4. Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, and M. Piattini, Generating Event Logs from Non-Process-Aware Systems Enabling Business Process Mining. Enterprise Information System Journal, 2011. 5(3): p. 301–335. Impact factor: 0.786.

JCR-5.Pérez-Castillo, R., I.G.-R.deGuzmán, and M. Piattini, Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems. Computer Standards & Interfaces, 2011. 33(6): p. 519-532. Impact factor: 0.825.

JCR-6.Pérez-Castillo, R., I.G.-R.deGuzmán, M. Piattini, and C. Ebert, Reengineering Technologies. IEEE Software Magazine, 2011. 28(6): p. 13-17. Impact factor: 1.511

JCR-7. Pérez-Castillo, R., J.A. Cruz-Lemus, I. García Rodríguez de Guzmán, and P. Mario, A Family of Case Studies on Business Process Mining using MARBLE.Journal of Systems and Software.86(6): p. 1370–1385.Impact factor: 1.277.

JCR-8.Pérez-Castillo, R., I. García Rodríguez de Guzmán, I. Caballero, and P. Mario, Software Modernization by Recovering Web Services from Legacy Databases. Journal of Software: Evolution and Process: Research and Practice. In Press (DOI: 10.1002/smr.1554). Impact factor: 0.606.

International Conferences (CORE A)

ITA-1.Pérez-Castillo, R., L. Sánchez-González, F. García, M. Piattini, and I. García Rodríguez de Guzmán, Obtaining Thresholds for the Effectiveness of Business Process Mining, in 5th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'11). 2011, IEEE ComputerSociety: Banff, Alberta, Canada. p. 453-462.

ITA-2. Pérez-Castillo, R., M. Fernández-Ropero, I. García Rodríguez de Guzmán, and M. Piattini, MARBLE. A Business Process Archeology Tool, in 27th IEEE International Conference on Software Maintenance (ICSM'11). 2011, IEEE Computer Society: Williamsburg, Virginia, USA. p. 578-581.

International Conferences (CORE B)

ITB-1. Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán and P. Mario (2012). Improving Event Correlation for Non-Process Aware Information Systems. 7th

International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE'12). Wroclaw, Poland: In Press.

ITB-2. Fernández-Ropero, M., R. Pérez-Castillo, I. Caballero, and M. Piattini, Quality-Driven Business Process Refactoring, in International Conference on Business Information Systems (ICBIS 2012). 2012: Paris, France.In Press.

ITB-3. Fernández-Ropero, M., R. Pérez-Castillo, and M. Piattini, Refactoring Business Process Models: A Systematic Review, in 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012). 2012, Springer-Verlag: Wrocław, Poland.In Press.

ITB-4.Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, and M. Piattini, Integrating Event Logs into KDM Repositories, in 27th Annual ACM Symposium on Applied Computing (SAC'12). 2012, ACM: Riva del Garda (Trento), Italy. p. 1095-1102.

ITB-5.Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, M. Piattini, and Á.S. Places, An Empirical Comparison of Static and Dynamic Business Process Mining, in 26th Annual ACM Symposium on Applied Computing (SAC'11). 2011, ACM: TaiChung, Taiwan. p. 269-276.

ITB-6.Pérez-Castillo, R., I. García-Rodríguez de Guzmán, O. Ávila-García, and M. Piattini, Business Process Patterns for Software Archeology, in 25th Annual ACM Symposium on Applied Computing (SAC'10). 2010, ACM: Sierre, Switzerland. p. 165-166.

ITB-7. Pérez-Castillo, R., I. García Rodríguez de Guzmán, O. Ávila-García, and P. Mario. On the use of ADM to Contextualize Data on Legacy Source Code for Software Modernization.in 16 th Working Conference on Reverse Engineering (WCRE). 2009. Lille, France: IEEE Computer Society.

ITB-8.Pérez-Castillo, R., I. García-Rodríguez de Guzmán, I. Caballero, M. Polo, and M. Piattini, PRECISO: A Reengineering Process and a Tool for Database Modernisation through Web Services in 24th Annual ACM Symposium on Applied Computing (SAC'09). 2009: Hawaii, USA. p. 2126-2133.

ITB-9. Pérez-Castillo, R., I. García-Rodríguez de Guzmán, I. Caballero, M. Polo, and M. Piattini. PRECISO: A Reverse Engineering Tool to Discover Web Services from Relational Databases. in 16 th Working Conference on Reverse Engineering (WCRE'09). 2009. Lille, France: IEEE ComputerSociety.

ITB-10.Pérez-Castillo, R., I. García Rodríguez de Guzmán, I. Caballero, M. Polo, and M. Piattini. PRECISO: A Reengineering Process and a Tool for Database Modernisation through Web Services in 24th Annual ACM Symposium on Applied Computing. Track on Service Oriented Architectures and Programming (SOAP 2009). 2009. Waikiki Beach, Honolulu, Hawaii, U.S.A.: ACM.

ITB-11. Pérez-Castillo, R., I. Caballero, I. García Rodríguez de Guzmán, M. Polo, M. Piattini, and E. Verbo. Establishing Trust Networks based on Data Quality Criteria for Selecting Data Suppliers. in 11th International Conference on Enterprise Information Systems (ICEIS'09). 2009. Milan, Italy: INSTICC.

ITB-12. Blanco, C., R. Pérez-Castillo, A. Hernández, E. Fernández-Medina, and J. Trujillo, Towards a modernization process for Secure Data Warehouses, in 11th International Conference on Data Warehousing and Knowledge Discovery (DaWaK). 2009, Springer-Verlag Berlin Heidelberg: Linz, Austria. p. 24-35.

Other International Conferences

ITC-1. Pérez-Castillo, R., I. García Rodríguez de Guzmán and P. Mario (2012). Database Schema Elicitation to Modernize Relational Databases. 14th International Conference on Enterprise Information Systems (ICEIS'12). In Press.

ITC-2. Fernández-Ropero, M., R. Pérez-Castillo, B. Weber, and M. Piattini, Empirical Assessment of Business Model Transformations Based on Model Simulation, in 5th International Conference on Model Transformation (ICMT 2012), Z. Hu and J.D. Lara, Editors. 2012, Springer-Verlag Berlin Heidelberg 2012: Prague, Czech Republic. p. 137–151.

ITC-3. Pérez-Castillo, R., I. G. R. d. Guzmán, F. García and M. Piattini (2012). A Practical Teaching Experience about Software Reengineering. 2nd World Conference of Educational Technology Researchers (WCETR'12). Kyrenia, North Cyprus, Procedia Journal (Elseiver): In Press.

ITC-4. Pérez-Castillo, R., I. García Rodríguez de Guzmán, and M. Piattini, Diagnosis of Software Erosion through Fuzzy Logic, in IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE'11). 2011, IEEE Computer Society: Paris, France. p. 49-56.

ITC-5. Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, and M. Piattini, Toward Obtaining Event Logs from Legacy Code.Business Process Management Workshops (BPI'10), 2010. LNBIP 66: p. 201–207.

ITC-6. Pérez-Castillo, R., I. García-Rodríguez de Guzmán, and M. Piattini, Implementing Business Process Recovery Patterns through QVT Transformations, in International Conference on Model Transformation (ICMT'10). 2010, Springer-Verlag: Málaga, Spain. p. 168-183.

ITC-7. Perez-Castillo, R., C. Guerra-Garcia, M. Piattini, and M.A. Moraga. Developing Data Quality Aware Applications.in 1st International Workshop on Data Quality for Software Engineering (DQ4SE) in conjuction with the 9th International Conference on Quality Software (QSIC). 2009. Jeju, Korea: IEEE Computer Society.

ITC-8.Pérez-Castillo, R., I. García-Rodríguez de Guzmán, O. Ávila-García, and M. Piattini, MARBLE: A Modernization Approach for Recovering Business Processes from Legacy Systems, in International Workshop on Reverse Engineering Models from Software Artifacts (REM'09). 2009, Simula Research Laboratory Reports: Lille, France. p. 17-20.

International Book Chapters

CHP-1.Pérez-Castillo, R., I. García Rodríguez de Guzmán, and M. Piattini, Architecture-Driven Modernization, in Modern Software Engineering Concepts and Practices: Advanced Approaches, A.H. Dogru and V. Bier, Editors. 2011, IGI Global. p. 75-103.

CHP-2.Pérez-Castillo, R., I. García Rodríguez de Guzmán, and M. Piattini, Model-Driven Reengineering, in Emerging Technologies for the Evolution and Maintenance of Software Models, J. Rech and C. Buns, Editors. 2012, IGI Global. p. 200-229.

Latin American Journals

LAJ-1. Pérez-Castillo, R., I. García Rodríguez de Guzmán, O. Ávila-García, A. Estévez, and M. Piattini, Modernización de Bases de Datos Relacionales mediante KDM. CUORE. Círculo de Usuarios Oracle de España, 2009. 40: p. Vivat Academia.

LAJ-2.Verbo, E., I. Caballero, R. Pérez-Castillo, C.Calero, and M. Piattini, MEPLAMECAL: A Methodology Based on ISO/IEC 15939 to Elaborate Data Quality Measurement Plans. IEEE Latin AmericaTransactions, 2009. 7(3): p. 361-368.

LAJ-3. Verbo, E., I. Caballero, R. Pérez-Castillo, C. Calero, and M. Piattini, Una metodología basada en ISO/IEC 15939 para la elaboración de planes de medición de calidad de datos. Novática, 2009. 197(enero-febrero 2009): p. 56-62.

Latin American Conferences

LAC-1. Pérez-Castillo, R., F. García, I. G. R. d. Guzmán and M. Piattini (2012). La Reingeniería como tópico en la Docencia de la Ingeniería del Software: una Experiencia Práctica. XVIII Jornadas de la Enseñanza Universitaria de la Informática. Ciudad Real, Spain: In Press.

LAC-2. Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, and M. Piattini, Modernización Software mediante Descubrimiento de Vistas de Procesos de Negocio, in XVI Jornadas de Ingeniería del Software y Bases de Datos (JISBD'11) - Track Desarrollo Software Dirigido por Modelos (DSDM). 2011: A Coruña, Spain. p. In Press.

LAC-3. Fernández-Ropero, M., R. Pérez-Castillo, I. García Rodríguez de Guzmán, and M. Piattini, MARBLE, in XVI Jornadas de Ingeniería del Software y Bases de Datos

(JISBD'11) - Track Desarrollo Software Dirigido por Modelos (DSDM). 2011: A Coruña, Spain. p. In Press.

LAC-4. Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, and M. Piattini, ModernizingLegacy Systems throughRuntimeModels, in VII Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM'10). 2010: Valencia, Spain. p. 78-87.

LAC-5. Pérez-Castillo, R., I. García-Rodríguez de Guzmán, O. Ávila-García, and M. Piattini. MARBLE: Un enfoque ADM para la obtención de Procesos de Negocio. in VI Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM'09) en XIV Jornadas de Ingeniería del Software y Bases de Datos (JISBD'09). Vol. 3, No. 2. 2009. San Sebastián, España.

LAC-6. Pérez-Castillo, R., I. García-Rodríguez de Guzmán, I. Caballero, M. Polo, and M. Piattini. PRECISO: Proceso de Reingeniería para la Extracción y Creación Instantánea de Servicios Web a partir de Orígenes de Datos. in IV Jornadas Científico Técnicas en Servicios Web y SOA. JSWEB 2008. 2008. Sevilla, Spain: San Antonio, Sdad. Coop. And.

LAC-7. García-Rodríguez de Guzmán, I., M. Polo, M. Piattini, and R. Pérez-Castillo. Un enfoque ADM para la Reingeniería de Bases de Datos Relacionales hacia Servicios Web in VII Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento (JIISIC 08). 2008. Guayaquil - Ecuador.

LAC-8. Verbo, E., I. Caballero, R. Pérez-Castillo, C. Calero, and M. Piattini. Una metodología basada en ISO/IEC 15939 para la elaboración de planes de medición de calidad de datos in XIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2008). 2008. Gijón, Spain.

6.5.2. Submitted Papers

As a result of the development of this Thesis, some research works have recently been submitted for their considered in journals, conferences and books.

SUB-1. Pérez-Castillo, R., M. Piattini, D. Caivano, T. Baldassarre and G. Visaggio "A Clustering Technique for Improving Business Process Archeology." IEEE Transactions on Software Engineering: Submitted.

SUB-2.Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, P. Mario, and J. Pinggera, Assessing Event Correlation in Non-Process-Aware Information Systems. Software and Systems Modeling Journal.Submitted.

SUB-3.Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán, and P. Mario, Concept Location Modeling Through Business Process Views. International Journal of CooperativeInformation Systems. Submitted.

SUB-4. Pérez-Castillo, R., I. García Rodríguez de Guzmán, and P. Mario, Modernización dirigida por modelos, in Desarrollo de Software Dirigido por Modelos: Conceptos, Métodos y Herramientas, F. García, Editor. p. Submitted.

SUB-5. Pérez-Castillo, R. and P. Mario (2012). "Model-Driven Reverse Engineering of Open Source Systems: A Global Software Development Project." Science on Computer Programming Submitted.

6.5.3. Relationship between Publications and Research Themes

In order to understand which research issue each publication deals with, Table 3 provides a matrix of the relationships between publications and research issues or themes of the PhD Thesis.

6.5.4. References to our Research Work

As a part of the result support, this section provides external references to the research publications obtained from the development of this Thesis:

REF-1.Xiaa, Y., W. Sub, R.Y.K. Lauc, and Y. Liua, Discovering latent commercial networks from online financial news articles. Enterprise Information System Journal (Taylor & Francis), 2011: p. In Press.

REF-2.Saroop, S. and M. Kumar, Comparative Analysis of Data warehouse Design Approaches from Security Perspectives.International Journal of Computer Trends and Technology (IJCTT), 2011. 1(3): p. 205-210.

REF-3.Rodrıguez-Echeverrıa, R., J.M. Conejero, P.J. Clemente, J.C. Preciado, and F. Sánchez-Figueroa, Modernization of Legacy Web Applications into Rich Internet Applications. 2011.

REF-4. Óscar Sánchez, R., C. Jesus Sánchez, and M. Jesus Garcia, Reverse Engineering of Event Handlers of RAD-Based Applications, in 18th Working Conference on Reverse Engineering (WCRE'2011). 2011, IEEE Computer Society: Limerick, Ireland. p. 293-302.

REF-5.MirshamsShahshahani, P., Extending the Knowledge Discovery Metamodel to Support Aspect-Oriented Programming. 2011, Concordia University.p. 79.

REF-6. Liu, W., Y.Y. Du, and C. Yan, Soundness preservation in composed logical time workflow nets. Enterprise Information System Journal (Taylor & Francis), 2011: p. In Press.

REF-7. Jensen, J. and M.G. Jaatun, Security in Model Driven Development: A Survey, in Sixth International Conference on Availability, Reliability and Security (ARES), 2011 2011, IEEE Computer Society: Viena, Austria. p. 704-709.

Table 3. Relationship between publications and research issues of the PhD Thesis

	State of the Art	MARBLE	Static Technique	Dynamic Technique	Supporting Tool	Event Correlation	Process Views	Process Refactoring	Empirical Assessment
JCR-1		♦							♦
JCR-2		♦	♦						♦
JCR-3				♦					♦
JCR-4				♦	♦	♦			♦
JCR-5	♦								♦
JCR-6	♦								♦
JCR-7		♦							♦
JCR-8	♦								
ITA-1									♦
ITA-2					♦				
ITB-1				♦		♦	♦		
ITB-2								♦	
ITB-3	♦							♦	
ITB-4				♦					
ITB-5			♦	♦					♦
ITB-6			♦						
ITB-7			♦						
ITB-8	♦								
ITB-9	♦								
ITB-10	♦								
ITB-11	♦								
ITB-12	♦								
ITC-1	♦								
ITC-2				♦	♦				
ITC-3	♦								
ITC-4	♦								♦
ITC-5				♦		♦			
ITC-6			♦						
ITC-7	♦								
ITC-8		♦							
CHP-1	♦	♦							
CHP-2	♦								
LAJ-1	♦								
LAJ-2	♦								
LAJ-3	♦								
LAC-1	♦								
LAC-2							♦		
LAC-3					♦				
LAC-4				♦					
LAC-5		♦							
LAC-6	♦								
LAC-7	♦								
LAC-8	♦								

REF-8. Collado, J.M., V. de Castro, M.L. Sanz, and E. Marcos, Marco para la Modernización de Sistemas de Información basado en Modelos: Aplicación a un Caso Práctico (transmisión de dominios. es), in XVI Jornadas de Ingeniería del Software y Bases de Datos (JISBD'11). 2011, SISTEDES: A Coruña, Spain. p. In Press.

REF-9.Velmurugan, K. and M.A.M. Mohamed, A Model driven Approach for Migrating from Legacy Software Systems to Web Services, in Indian Workshop in Reverse Engineering (WIRE'10). 2010: Mysuru, Karnataka, India.

REF-10. Rodríguez, A., A. Caro, and E. Fernández-Medina, Towards framework definition to obtain secure business process from legacy information systems, in Proceeding of the first international workshop on Model driven service engineering and data quality and security. 2009, ACM: Hong Kong, China. p. 17-24.

REF-11.Tan, W., L. Li, W. Xu, F. Yang, C. Jiang, L. Yang and J. Choi (2012). "A role-oriented service system architecture for enterprise process collaboration." Computers and Operations Research 39(8): 1893-1900.

REF-12.Frey, S., W. Hasselbring and B. Schnoor (2012). "Automatic conformance checking for migrating software systems to cloud infrastructures and platforms." Journal of Software Maintenance and Evolution: Research and Practice: In Press (DOI: 10.1002/smr.1582).

REF-13.Peslak, A. (2012). "An analysis of critical information technology issues facing organizations." Industrial Management & Data Systems 112(5): In Press.

REF-14.Vela, B., C. Blanco, E. Fernández-Medina and E. Marcos (2012). "A practical application of our MDD approach for modeling secure XML data warehouses."Decision Support Systems 52(4): 899-925.

REF-15.Ramon, Ó. S., J. S. Cuadrado and J. G. Molina (2011).Reverse Engineering of Event Handlers of RAD-Based Applications. 18th Working Conference on Reverse Engineering (WCRE'11). Lero, Limerick, Ireland, IEEE Computer Society: 293-302.

REF-16.Wimmer, M., A. Kusel, W. Retschitzegger, J. Schönböck, W. Schwinger, J. S. Cuadrado, E. Guerra and J. de Lara (2011). "5th International Workshop on Multi-Paradigm Modeling (MPM 2011)." Electronic Communications of the EASST 42.

REF-17. Panjkov, Z., M. Katona, D. Spasojevic and Z. Pele (2012). A Case Study in Software Reengineering for a DSP-based System on a Chip: Adaptation of Dolby Virtual Speaker. 19th International Conference and Workshops on Engineering of Computer-Based Systems. Novi Sad, Serbia, IEEE Computer Society: 303-307.

References

REFERENCES

Avison, D. E., F. Lau, M. D. Myers and P. A. Nielsen (1999). "Action research." Communications of the ACM 42(1): 94-97.

Bender, A. P., D. L. Parker, R. A. Johnson, W. K. Scharber, A. N. Williams, M. C. Marbury and J. S. Mandel (1989). "Minnesota highway maintenance worker study: cancer mortality." American journal of industrial medicine 15(5): 545-556.

Brereton, P., B. Kitchenham, D. Budgen and Z. Li (2008). Using a protocol template for case study planning. Evaluation and Assessment in Software Engineering (EASE'08). Bari, Italia: 1-8.

Cai, Z., X. Yang and W. Wang (2009). Business Process Recovery for System Maintenance - An Empirical Approach. 25 th International Conference on Software Maintenance (ICSM'09). Edmonton, Alberta, Canada, IEEE Computer Society: 399-402.

Canfora, G., M. Di Penta and L. Cerulo (2011). "Achievements and challenges in software reverse engineering." Commun. ACM 54(4): 142-151.

Castellanos, M., K. A. d. Medeiros, J. Mendling, B. Weber and A. J. M. M. Weitjers (2009). Business Process Intelligence. Handbook of Research on Business Process Modeling. J. J. Cardoso and W. M. P. van der Aalst, Idea Group Inc.: 456-480.

Chikofsky, E. J. and J. H. Cross (1990). "Reverse Engineering and Design Recovery: A Taxonomy." IEEE Softw. 7(1): 13-17.

Davenport, T. H. (1993). "Need radical innovation and continuous improvement? Integrate process reengineering and TQM." Strategy & Leadership Journal 21(3): 6-12.

Di Francescomarino, C., A. Marchetto and P. Tonella (2009). Reverse Engineering of Business Processes exposed as Web Applications. 13th European Conference on Software Maintenance and Reengineering (CSMR'09). Fraunhofer IESE, Kaiserslautern, Germany, IEEE Computer Society: 139-148.

Ghose, A., G. Koliadis and A. Chueng (2007). Process Discovery from Model and Text Artefacts. IEEE Congress on Services (Services'07), Salt Lake City, UT.

Günther, C. W. and W. M. P. van der Aalst (2007). "A Generic Import Framework for Process Event Logs." Business Process Intelligence Workshop (BPI'06) LNCS 4103: 81-92.

Hammer, M. and J. Champy (1993). Reengineering the Corporation: A Manifesto for Business Revolution, HarperBusiness.

Hedges, L. V. and I. Olkin (1985). Statistical Methods for Meta-Analysis, Academia Press.

Heuvel, W.-J. v. d. (2006). Aligning Modern Business Processes and Legacy Systems: A Component-Based Perspective (Cooperative Information Systems), The MIT Press.

Ingvaldsen, J. E. and J. A. Gulla (2008). "Preprocessing Support for Large Scale Process Mining of SAP Transactions." Business Process Intelligence Workshop (BPI'07) LNCS 4928: 30-41.

ISO/IEC (2006). ISO/IEC 14764:2006. Software Engineering -- Software Life Cycle Processes -- Maintenance. http://www.iso.org/iso/catalogue_detail.htm?csnumber=39064, ISO/IEC.

ISO/IEC (2009). ISO/IEC DIS 19506. Knowledge Discovery Meta-model (KDM), v1.1 (Architecture-Driven Modernization). http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2=080&ics3=&csnumber=32625, ISO/IEC: 302.

Jeston, J., J. Nelis and T. Davenport (2008). Business Process Management: Practical Guidelines to Successful Implementations. NV, USA, Butterworth-Heinemann (Elsevier Ltd.).

Lehman, M. M. (1984). "Program evolution." Information Processing & Management 20(1-2): 19-36.

Lewis, G. A., D. B. Smith and K. Kontogiannis (2010). A Research Agenda for Service-Oriented Architecture (SOA): Maintenance and Evolution of Service-Oriented Systems, Software Engineering Institute: 40.

Medeiros, A. K., A. J. Weijters and W. M. Aalst (2007). "Genetic Process Mining: An Experimental Evaluation." Data Min. Knowl. Discov. 14(2): 245-304.

Medeiros, A. K. A., B. F. van Dongen, v. d. AalstW.M.P. and A. J. M. M. Weijters (2004). "Process Mining: Extending the alpha-algorithm to Mine Short Loops." BETA Working Paper Series WP(113).

Miller, J. (2000). "Applying Meta-Analytical Procedures to Software Engineering Experiments." Journal of Systems and Software 54: 29-39.

Moody, D. L. (2000). Building links between IS research and professional practice: improving the relevance and impact of IS research. Proceedings of the twenty first international conference on Information systems. Brisbane, Queensland, Australia, Association for Information Systems: 351-360.

Motahari-Nezhad, H. R., R. Saint-Paul, F. Casati and B. Benatallah (2011). "Event Correlation for Process Discovery From Web Service Interaction Logs." The VLDB Journal 20(3): 417-444.

OMG (2008). QVT. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. http://www.omg.org/spec/QVT/1.0/PDF, OMG.

OMG (2009). Business Process Model and Notation (BPMN) 2.0, Object Management Group: 496.

Paradauskas, B. and A. Laurikaitis (2006). "Business Knowledge Extraction from Legacy Information Systems." Journal of Information Technology and Control 35(3): 214-221.

Paradauskas, B. and A. Laurikaitis (2006). "Business Knowledge Extraction from Legacy Informations Systems." Information Technology and Control 35(3): 214-221.

Pérez-Castillo, R., I. G.-R. de Guzmán and M. Piattini (2011). "Business Process Archeology using MARBLE." Information and Software Technology 53: 1023–1044.

Pérez-Castillo, R., I. García-Rodríguez de Guzmán, O. Ávila-García and M. Piattini (2010). Business Process Patterns for Software Archeology. 25th Annual ACM Symposium on Applied Computing (SAC'10). Sierre, Switzerland, ACM: 165-166.

Pérez-Castillo, R., I. García-Rodríguez de Guzmán and M. Piattini (2010). Implementing Business Process Recovery Patterns through QVT Transformations. International Conference on Model Transformation (ICMT'10). Málaga, Spain, Springer-Verlag. LNCS 6142: 168-183.

Pérez-Castillo, R., B. Weber, I. García Rodríguez de Guzmán and M. Piattini (2010). "Toward Obtaining Event Logs from Legacy Code." Business Process Management Workshops (BPI'10) Lecture Notes in Business Information Processing ((LNBIP 66 - Part 2)): 201–207.

Pickard, L. M. (2004). Combining Empirical Results in Software Engineering, University of Keele.

Rozinat, A., I. S. M. D. Jong, C. W. G\, \#252, nther and W. M. P. V. D. Aalst (2009). "Process mining applied to the test process of wafer scanners in ASML." Trans. Sys. Man Cyber Part C 39(4): 474-479.

Runeson, P. and M. Höst (2009). "Guidelines for Conducting and Reporting Case Study Research in Software Engineering." Empirical Softw. Eng. 14(2): 131-164.

Santos, P. S. M. d. and G. H. Travassos (2009). Action research use in software engineering: An initial survey. Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, IEEE Computer Society: 414-417.

Sneed, H. M. (2005). Estimating the Costs of a Reengineering Project, IEEE Computer Society.

Soares Cruzes, D., T. Dybå, P. Runeson and M. Höst (2011). Case studies synthesis: brief experience and challenges for the future. 5th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'11). Banff, Alberta, Canada, IEEE Computer Society: 46-55.

The Standish Group (2010). Modernization. Clearing a pathway to sucess, The Standish Group International, Inc.

van der Aalst, W., H. Reijers and A. Weijters (2007). "Business Process Mining: An Industrial Application." Information Systems 32(5): 713-732.

van der Aalst, W. and A. J. M. M. Weijters (2005). Process Mining. Process-aware information systems: bridging people and software through process technology. M. Dumas, W. van der Aalst and A. Ter Hofstede, John Wiley & Sons, Inc.: 235-255.

Van der Aalst, W. M. P., B. F. Van Dongenm, C. Günther, A. Rozinat, H. M. W. Verbeek and A. J. M. M. Weijters (2009). ProM : The Process Mining Toolkit. 7th International Conference on Business Process Management (BPM'09) - Demonstration Track. Ulm, Germany, Springer-Verlag. 489: 1-4.

Weske, M. (2007). Business Process Management: Concepts, Languages, Architectures. Leipzig, Germany, Springer-Verlag Berlin Heidelberg.

Wood-Harper, T. (1985). "Research methods in information systems: using action research." Research methods in information systems: 169-191.

Yin, R. K. (2003). Case Study Research. Design and Methods. London, Sage.

Zou, Y. and M. Hung (2006). An Approach for Extracting Workflows from E-Commerce Applications. Proceedings of the Fourteenth International Conference on Program Comprehension, IEEE Computer Society: 127-136.