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.

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

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

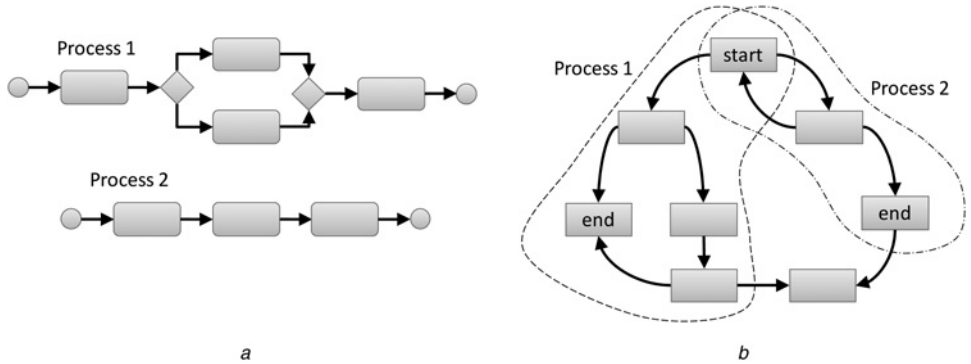

Fig. 1 *Comparison between PAIS and traditional information systems*
a Process-aware information system
b Traditional information system

doi: 10.1049/iet-sen.2010.0103

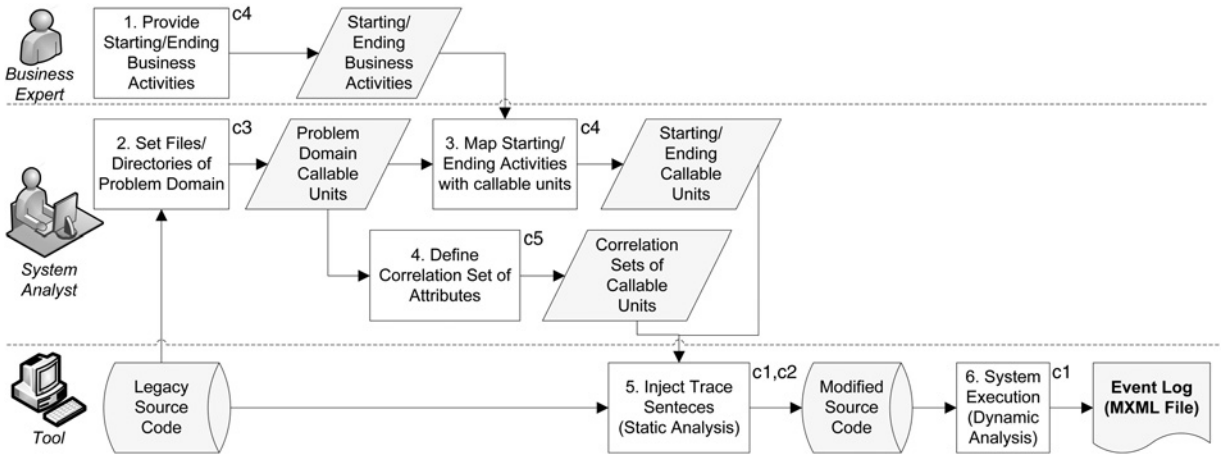

Fig. 2 *Overall process carried out by means of the proposed technique*

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

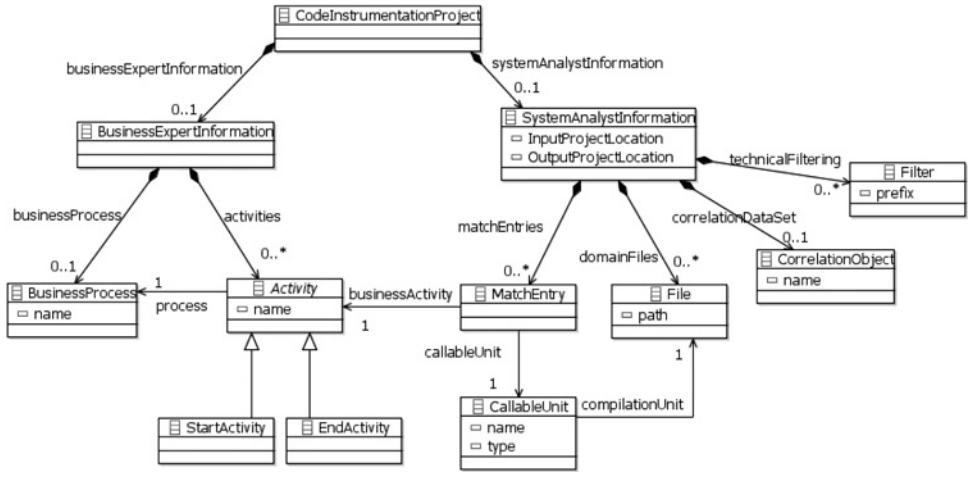

Fig. 3 *Metamodel to formalise and collect all the information of the code instrumentation stage*

```
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

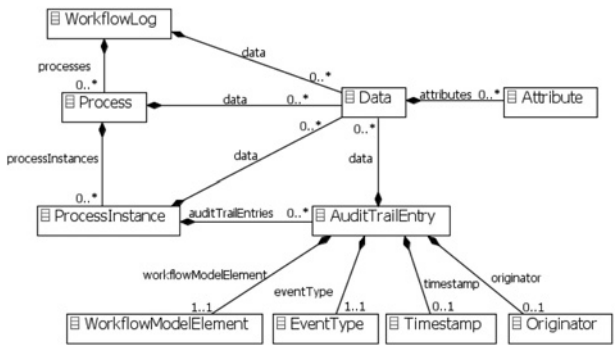

Fig. 5 *MXML metamodel used to represent runtime models in MARBLE*

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

doi: 10.1049/iet-sen.2010.0103

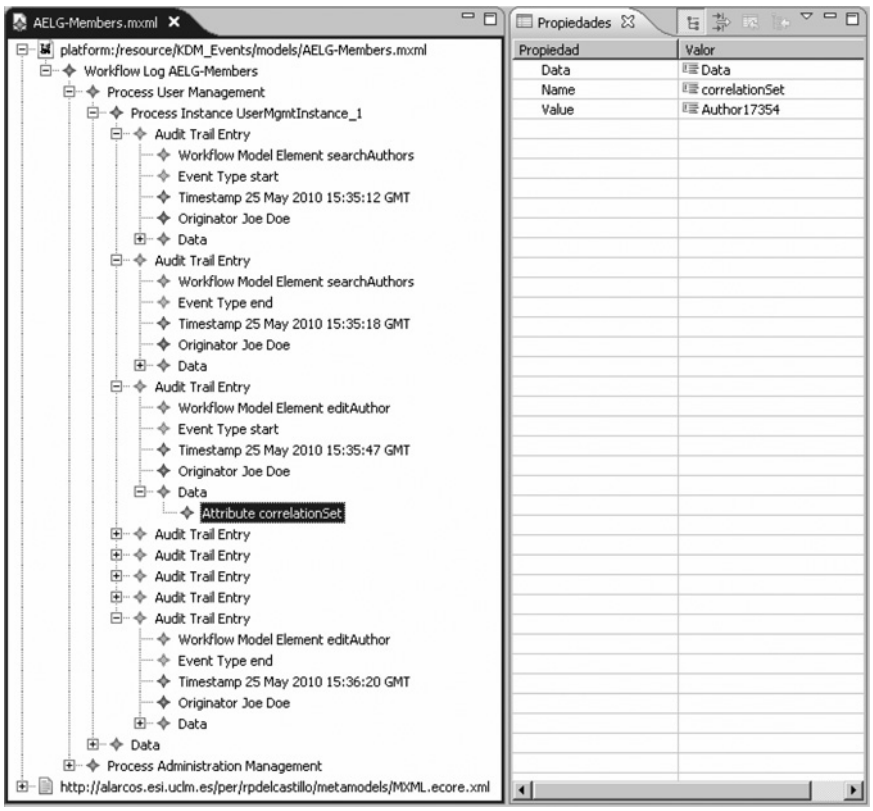

Fig. 6 *MXML model obtained by applying the technique with an author management system*

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)

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>,

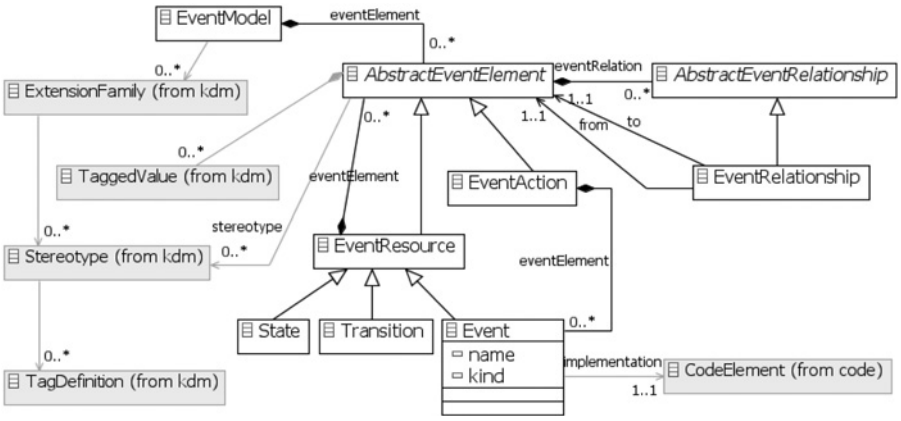

Fig. 7 *Event metamodel package in the KDM standard*

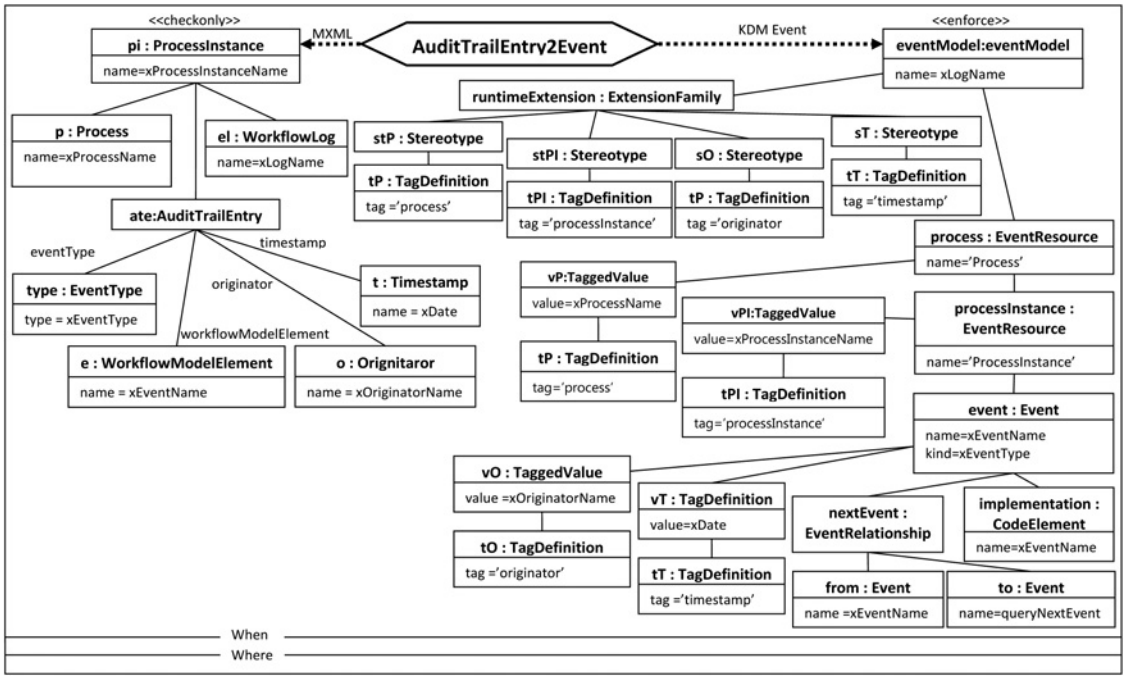

Fig. 8 *QVT relation 'auditTrailEntry2Event' to transform MXML models into KDM event model*

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.

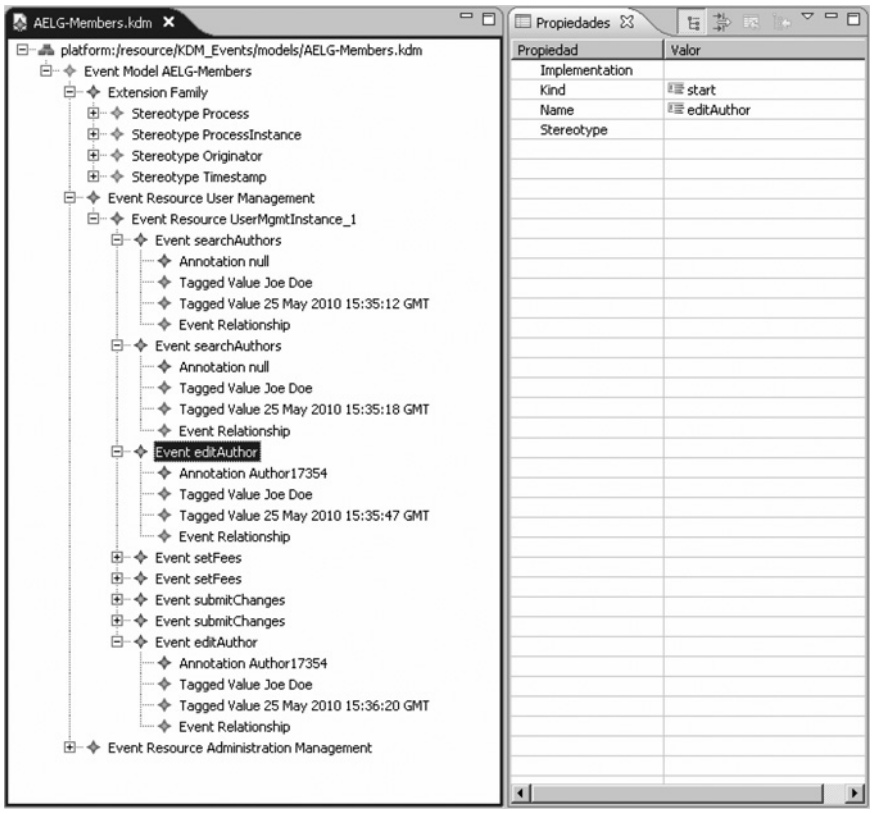

Fig. 9 *KDM event model obtained from the MXML model concerning an author management system*

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

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

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.

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.

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″

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

doi: 10.1049/iet-sen.2010.0103

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

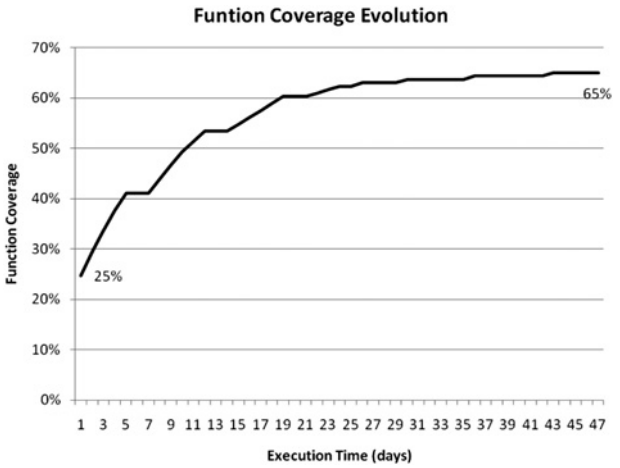

Fig. 11 *Function coverage evolution during system execution*

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.

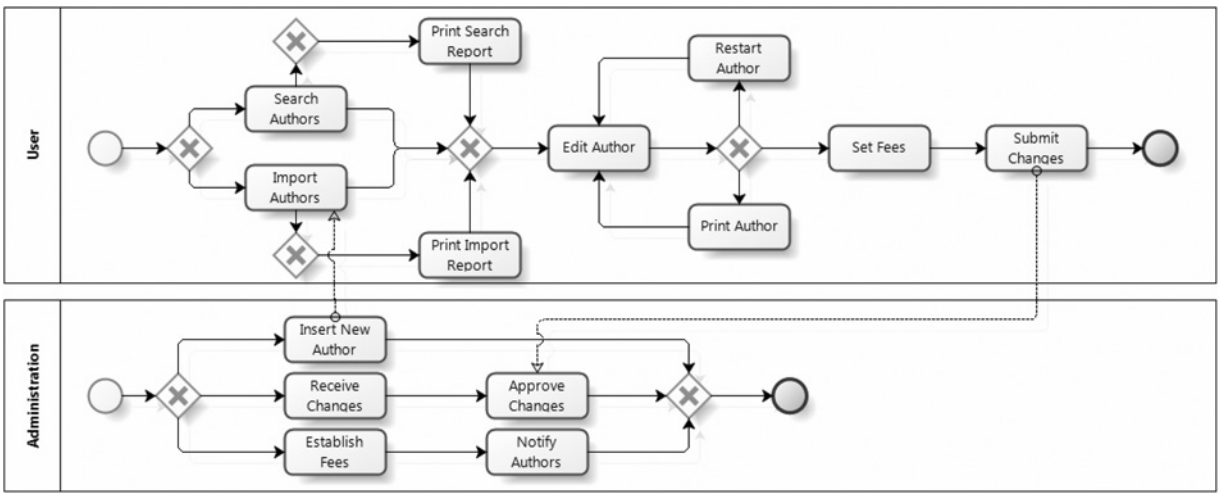

Fig. 10 *Final 'author management' business process*

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

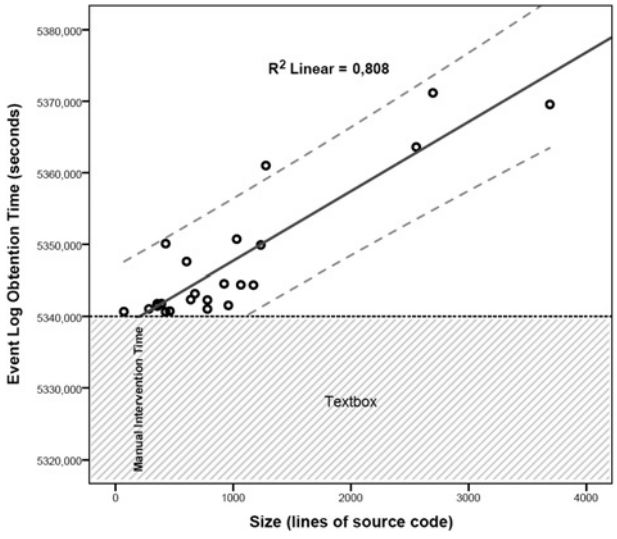

Fig. 12 *Size/time scatter chart*

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

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

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