Empirical Assessment of Business Model Transformations Based on Model Simulation

María Fernández-Ropero[1], Ricardo Pérez-Castillo[1], Barbara Weber[2], and Mario Piattini[1]

[1] Instituto de Tecnologías y Sistemas de la Información, University of Castilla-La Mancha
Paseo de la Universidad 4 13071, Ciudad Real, Spain
{marias.fernandez,ricardo.pdelcastillo,mario.piattini}@uclm.es
[2] University of Innsbruck
Technikerstraße 21a, 6020, Innsbruck, Austria
barbara.weber@uibk.ac.at

Abstract. Business processes are recognized by organizations as one of the most important intangible assets, since they let organizations improve their competitiveness. Business processes are supported by enterprise information systems, which can evolve over time and embed particular business rules that are not present anywhere else. Thus, there are many organizations with inaccurate business processes, which prevent the modernization of enterprise information systems in line with the business processes that they support. Therefore, business process mining techniques are often used to retrieve reliable business processes from the event logs recorded during the execution of enterprise systems. Unfortunately, such event logs are represented with purpose-specific notations such as Mining XML and still don't apply the recent software modernization standard: ISO 19506 (KDM, Knowledge Discovery Metamodel). This paper presents an exogenous model transformation between these two notations. The main advantage is that process mining techniques can be effectively reused within software modernization projects according to the standard notation. This paper is particularly focused on the empirical evaluation of this transformation by simulating different kinds of business process models and several event logs with different sizes and configurations from such models. After analyzing all the model transformation executions, the study demonstrates that the transformation can provide suitable KDM models in a linear time in accordance with the size of the input models.

Keywords: Business Processes, Event Logs, Knowledge Discovery Metamodel, Model Simulation.

1 Introduction

Most companies recognize business processes as a valuable asset to carry out their daily operation with the aim of achieving their business goals [1]. Business processes management helps companies to continuously adapt their operation in order to maintain their degree of competitiveness.

Most parts of business processes are automatically supported by means of enterprise information systems [2]. These information systems unfortunately undergo software erosion overtime as a result of uncontrolled maintenance, and they become Legacy Information Systems (LIS). LIS embed much business knowledge that is not present anywhere else, which may imply that the business process representations of a company are misaligned with the actual business processes.

Software modernization is a suitable solution to address software erosion problems. Software modernization is the concept of evolving LIS with a focus on all aspects of the current system's architecture and the ability to transform current architectures into target architectures [3]. Software modernization improves the Return on Investment (ROI) by extending the lifecycle of systems, since it advocates preserving the embedded business knowledge. Business process mining techniques facilitate the preservation of business knowledge, since such techniques retrieve the actual, embedded business processes [4].

Business process mining techniques work with event logs recorded from the system execution, which represent the sequence of business activities executed by an enterprise system. Event logs models are often represented according to the Mining XML (MXML) metamodel [5]. These event logs represented according to MXML are suitable for most of the process mining techniques but they are not to be used in whole software modernization projects. For example, the discovered business processes model cannot have additional information about relationships between source code elements and the respective discovered business activities. This kind of information is necessary to understand and modernize LIS in line with the actual business processes supported by them [6].

Moreover, software modernization advocates the usage of the Knowledge Discovery Metamodel (KDM), which was recognized as the standard ISO 19506 [7], to represent different legacy software artifacts. KDM is organized into various orthogonal concerns (metamodel packages) that are in turn organized in different abstraction layers. The KDM event package allows representing event models alternatively to MXML.

This paper presents a declarative model transformation implemented using QVTr (Query/View/Transformation Relations) for transforming MXML event models into KDM event models [8]. The main advantage is that event logs transformed into KDM models can be integrated into software modernization processes so that synergies between event models and the remaining kinds of models (e.g., code model, database model, etc.) can be exploited together and in a homogeneous and standardized way.

This paper provides a formal experiment to empirically validate the model transformation. The experiment systematically simulates several event log models following different configurations. The study analyzes the effects of simulating factors (e.g., size of logs, complexity, etc.) on the efficiency of the model transformation. The result of this study demonstrates the scalability and suitability of the model transformation to be applied for obtaining KDM event models from MXML models in a linear time in accordance with the size and the complexity of the input model.

The remainder of this paper is organized as follows: Section 2 summarizes related work. Section 3 presents the model transformation under study. Section 4 describes the experiment based on model simulation. Section 5 provides the analysis and interpretation of results. Finally, Section 6 discusses conclusions and future work.

2 Related Work

Business process mining describes a family of a posteriori analysis techniques exploiting the information recorded in an event log [9]. Event logs sequentially record the business activities executed in process aware information systems. There are several works that use process mining dealing with the construction of business processes when there is no a priori business process model. For example, *Van der Aalst et al.* [10] propose the α-algorithm to discover the control flow of business processes from event logs. Similarly, *Madeiros et al.* [11] suggest a genetic algorithm for business process discovery.

Other proposals deal with the registration of event logs, e.g., *Ingvaldsen et al.* [12] focus on ERP (*Enterprise Resource Planning*) systems to obtain event logs from the SAP's transaction data logs. *Günther et al.* [13] provide a generic import framework for obtaining event logs from different kinds of systems. Other authors such as *Pérez-Castillo et al.* [14] propose an approach to obtain event logs by means of the injection of traces in legacy source code to enable the collection of event logs in non-process-aware systems.

All these proposals focus on the development and application of business process mining techniques. However, the mentioned approaches do not address the effective use of business processes to modernize legacy information systems being aligned with the actual business process. *Zou et al* [15] developed a framework that statically analyzes the legacy source code and applies a set of heuristic rules to recover the underlying business processes. Other works focus on recovering business processes by dynamically tracing the system execution driven by use cases (e.g., *Cai et al.* [16]), or driven by the users' navigation in graphical user interfaces (e.g., *di Francescomarino et al* [17]). The goal of these works is to obtain the actual, embedded business processes to be used during software modernization.

Unfortunately, all these works [15-17] propose *ad hoc* techniques that do not follow the KDM standard. As a consequence, the reuse as well as the scalability of these techniques to be applied to large and complex LIS is limited. In this sense, *Pérez-Castillo et al.* [8] present a preliminary method to integrate MXML event logs into KDM repositories, which is the starting point of this research. Nevertheless, this method has not been empirically validated, for example, through model simulation.

Model simulation is often applied in other research fields such as aerospace, healthcare, etc. Literature contains some proposals that use model simulation for empirically assessing model transformations. For instance, *Wong et al.* [18] use model simulation to empirically validate the translation of business process diagrams into executable BPEL (*Business Process Execution Language*) processes. *Syriani et al.* [19] use simulation to validate models of reactive systems such as modern computer games. *Biermann et al.* [20] propose simulation environments based on a model's concrete syntax definition for visual languages. The validation of the proposed model transformation follows a model simulation approach similar to such studies.

3 MXML to KDM Transformation

The proposed model transformation takes an MXML model and obtains an equivalent KDM model at the same abstraction level. MXML is the notation commonly used to represent event logs to be exploited in business process mining techniques [5] (see Fig. 1). An MXML model represents an individual log (*WorkflowLog*). The log consists of a set of business processes (*Process*) that collect, in turn, several instances of such processes (*ProcessInstance*). Each process instance represents a certain execution of a business process using particular data. For example, in a bank company, process could be different execution instances for different customers. Each process instance has a sequence of events (*AuditTrailEntry*). Each event consists of four elements: (i) the business activity executed (*WorkflowModelElement*); (ii) the type of the event (i.e., start or complete) (*EventType*); (iii) the user who started or completed the business activity (*Originator*); and finally (iv) the time when the event was recorded (*Timestamp*). All these elements can contain additional information through *Data* and *Attribute* elements.

On the other hand, The KDM Event metamodel (see Fig. 2) defines the *EventModel* metaclass to depict an event model in KDM. Each event model aggregates a set of event resources of a LIS (*EventResource*). Particularly, event resources can be states, transitions or events themselves (*Event*). Each event has two features: the *name* of the event and the *kind* (i.e., start or complete). Event resources and other elements can be related by means of event relationships (AbstractEventRelationship) which can depict next states, transitions, consumed events, etc. Moreover, the KDM event metamodel extends the KDM *action* package metamodel by defining a set of event actions that can be associated with event resources (see Fig. 2). For example, events that are produced by particular code elements can be represented with *ProducesEvent* elements. These elements contain references to pieces of source code (*CodeElement*) by means of the feature *implementation.* It enables the integration of KDM event models with the remaining of KDM models ensuring its appropriate usage in modernizations projects.

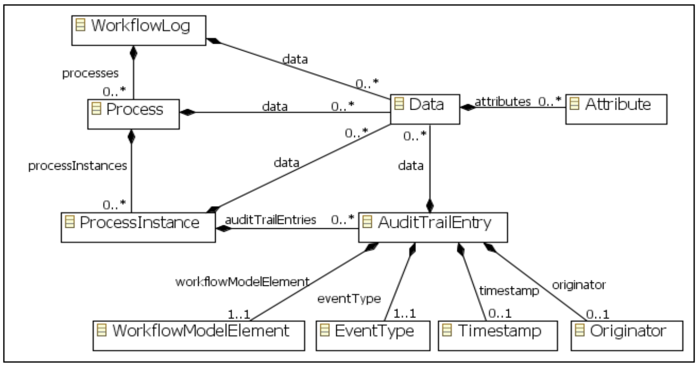

Fig. 1. The MXML Metamodel

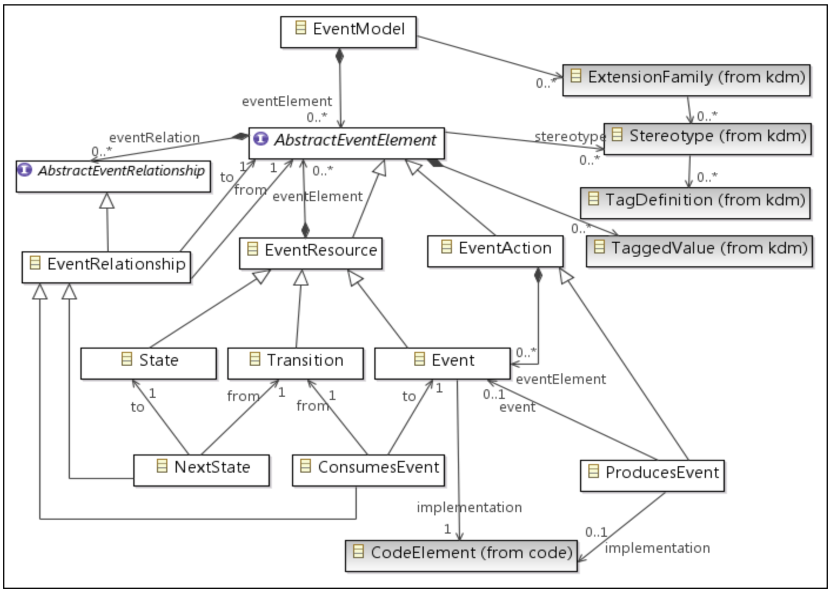

Fig. 2. The KDM event metamodel and extensions

3.1 Transformation Rules

The MXML to KDM model transformation consists of a set of eight declarative transformation rules. First of all, a KDM event model must be created from the MXML event log model (Rule 1). Events entail the key element of MXML event log models, thus, events must be transformed into the KDM event model (Rule 2). Furthermore, the information concerning the four components of an event (i.e., business activity name, type, originator and timestamp) must be represented in the KDM event model. Besides, the name and type of the events in the MXML models are represented in the KDM event models by using the features of the *Event* metaclass respectively (Rule 3). The information concerning the originator and timestamp cannot be directly represented in the KDM model according to the KDM event package. For this reason, the KDM event model must be extended with additional metaclasses so that it can support this information. Events (which represent executed business activities) are mapped to the pieces of source code that support those activities (Rule 4). This is possible because the KDM event model can be linked with other KDM models (e.g., the KDM code model) by means of the features *implementation* that link code elements (see Fig. 2).

Rule 1. *Each instance of the WorkflowLog metaclass is transformed into an instance of the EventModel metaclass in the output model.*

Rule 2. *Each instance of the AuditTrailEntry metaclass is transformed into an instance of the Event metaclass in the output model.*

Rule 3. *Instances of the WorkflowModelElement and EventType metaclass, belonging to an instance of the AuditTrailEntry metaclass, are respectively incorporated into the features 'name' and 'kind' of the respective instance of the Event metaclass (see Rule 2).*

Rule 4. *Instances of the Attribute metaclass with the name feature 'implementation' are transformed into instances of the CodeElement metaclass within the respective instance of the Event metaclass in the output model (see Rule 2).*

In order to represent all the information registered in a MXML model in the KDM model, the KDM event metamodel is extended by means of the *ExtensionFamily* metaclass, the standard extension mechanism of KDM (see highlighted metaclasses in Fig. 2). The extension family defines a set of stereotypes containing a set of tag definitions. Stereotypes define a wide concern while tag definitions specify the new elements that will be used in normal elements of the KDM event metamodel through tagged values. Tagged values allow changing or adjusting the meaning of those elements by associating a value with a previously defined tag. According to the extension mechanism, Rule 5 refines R1 by adding the extension family within the event model. The extension family has four stereotypes: <*process*>, <*processInstance*>, <*originator*> and <*timestamp*>. Event resources are tagged with <*process*> (Rule 6) and <*processInstance*> (Rule 7) to respectively collect business processes and their instances from event logs. Finally, both originator and timestamp are represented by incorporating tagged values to the respective event (Rule 8).

Rule 5. *An instance of the ExtensionFamily metaclass is created for each instance of the EventModel metaclass in the output model (see Rule 1). This instance contains four instances of the Stereotype metaclass. In turn, each Stereotype instance contains an instance of the TagDefiniton metaclass. The values of these four stereotypes are: <process>, <processInstance>, <originator> and <timestamp>.*

Rule 6. *Each instance of the Process metaclass is transformed in the output model into an instance of the EventResource metaclass with an instance of the TaggedValue metaclass. The tag feature of this instance links to the <Process> stereotype, and the value feature represents process name.*

Rule 7. *Each instance of the ProcessInstance metaclass is transformed in the output model into an instance of the EventResource metaclass with an instance of the TaggedValue metaclass. The tag feature of this instance links to the <ProcessInstance> stereotype, and the value feature represents the name of the business process instance.*

Rule 8. *Instances of the Originator and Timestamp metaclass are transformed into two instances of the TaggedValue metaclass which are added to the respective instance of the Event metaclass (see Rule 2). The instances of the TaggedValue metaclass respectively define their tag features as <Originator> and <Timestamp> stereotype, and their value feature with the name of the originator and timestamp registered in the input model.*

An executable version of the model transformation has been implemented using QVTr (Query/View/Transformation relations) [21], which provides a declarative and rule-based specification. Due to the space limitation this paper shows the *'auditTrailEntry2Event'* relation as an example (see Fig. 3). The full transformation is available online [22]. The *checkonly* domain of the relation is defined on instances of the *AuditTrailEntry* metaclass. This input domain checks the existence of the four elements in an event (i.e., the business activity, type, originator and timestamp). The input domain also evaluates the existence of the process instance, process and the event log where the *AuditTrailEntry* element belongs. The *enforce* domain creates an instance of the *Event* metaclass according to Rule 2. This event is created within the respective log, process and process instance. The originator and timestamp are added with the appropriate stereotype according to Rule 8. Finally, the *when* clause invokes the *'processInstance2eventResource'* to check, as a pre-condition, that the respective process instance was previously created by means of the invoked relation (see Fig. 3).

```
top relation auditTrailEntry2Event {
  xEventName : String;
  xEventType : String;
  xOriginatorName : String;
  xDate : String;
  xProcessInstanceName : String;
  xProcessName : String;
  xModelName : String;
  checkonly domain mxml ate : mxml::AuditTrailEntry {
    workflowModelElement = wme : mxml::WorkflowModelElement {
      name = xEventName
    },
    eventType = type : mxml::EventType {
      type = xEventType
    },
    originator = originator : mxml::Originator {
      name = xOriginatorName
    },
    timestamp = timestamp : mxml::Timestamp {
      date = xDate
    },
    processInstance = pi : mxml::ProcessInstance {
      name = xProcessInstanceName,
      process = p : mxml::Process {
        name = xProcessName,
        workflowLog = wl : mxml::WorkflowLog {
          name = xModelName
        }
      }
    }
  };
  enforce domain event eventModel:event::EventModel{
    name = xModelName,
    eventElement = eRes:event::EventResource {
      name = xProcessName,
      eventElement = eRes2:event::EventResource {
        name = xProcessInstanceName,
        eventElement = event : event::Event {
          name = xEventName,
          kind = xEventType,
          taggedValue = originatorTag : kdm::TaggedValue {
            tag = ot : kdm::TagDefinition {
              tag = 'Originator'
            },
            value = xOriginatorName
          },
          taggedValue = timestampTag : kdm::TaggedValue {
            tag = dt : kdm::TagDefinition {
              tag = 'Timestamp'
            },
            value = xDate
          },
          implementation = codeElement : code::CodeElement {
            name = xEventName
          }
        }
      }
    }
  };
  when {
    processInstance2eventResource (pi, eventModel);
  }
}
```

Fig. 3. The 'auditTrailEntry2Event' QVT relation

4 Experiment Description

This section presents one experiment to validate the proposed model transformation. The experiment is based on the formal protocol proposed by *Jedlitschka et al.* [23] for conducting and reporting empirical research in software engineering. According to this protocol, the following sections describe the research goal and questions, research hypothesis, variables, design and execution procedure, as well as the analysis procedure.

4.1 Research Goal and Questions

The main research goal of this experiment is the efficiency assessment of the transformation. In order to evaluate this property the experiment attempts to answer two research questions:

- RQ1: *Is the model transformation scalable to large MXML models?*
- RQ2: *Does the input model's complexity affect to the transformation performance?*

Firstly, scalability assessment (RQ1) is important to ensure the applicability of this transformation with large and complex event logs. Secondly, the study of side effects of the event log's complexity (RQ2) in the transformation performance is valuable to prove its feasibility with any kind of event log.

The study randomly simulates a set of event logs for assessing the research questions. Event logs are simulated through *Process Log Generator* (PLG) [24], a tool for the generation of business process models and simulation of different MXML logs (cf. Section 4.4).

4.2 Variables

A set of variables is defined for the assessment of the model transformation efficiency. There are two independent variables: (i) **Size**, which represents the number of events in the simulated MXML log; and (ii) **ECyM**, which represents the Extended Cyclomatic Metric (ECyM) of the MXML model [25]. ECyM determines how complicated the behavior of the model is, i.e., its complexity. The ECyM of a graph G with V vertices, E edges, and p connected components is: $ECyM = |E| - |V| + p$

The dependent variables of the study are two: (i) **Transformation Time**, which is the time spent on transforming a MXML model into a KDM model through the proposed model transformation; and (ii) **Performance,** which is the ratio between the size of the input model and the transformation time. This variable is normalized in the range [0, 1].

4.3 Research Hypothesis

In the case of RQ1, it is necessary to check if there is a linear relation between the size of the MXML model and the time of transformation through the proposed model transformation. To do this, the hypotheses are:

- $H_{RQ1,0}$: The size of the MXML model has a linear relation with the time of transformation.
- $H_{RQ1,1}$: The size of the MXML model has not a linear relation with the time of transformation

To answer RQ2 it is necessary to check if the input model's complexity affects the performance. To do this, the hypotheses are:

- $H_{RQ2,0}$: The ECyM of the log does not influence the performance.
- $H_{RQ2,1}$: The ECyM of the log influences the performance.

The goal of the statistical analysis is being able to accept these null hypotheses with an acceptable confidence level.

4.4 Design and Execution Procedure

The experiment evaluates the transformation model in several simulated event logs. The experiment's execution consists of the following steps:

1. The set of MXML logs are simulated using PLG. PLG allows users to obtain business processes with different sizes by defining the maximum number of nested branches. The study uses three sizes: 2, 3 and 4 maximum nested branches, which are respectively labeled as low, medium and high. Four business process models are created for each size. In turn, four event logs are simulated for each business process with different numbers of business process instances: 50, 100, 150, and 200. In total, 48 logs conform the sample to perform the experiment. For each log steps 2 to 3 are repeated.
2. The MXML log is analyzed for collecting relevant variables (i.e., number of events, ECyM, etc.).
3. The MXML is transformed into a KDM event model. The transformation is executed through *Medini QVT* [26], a model transformation engine supporting QVTr. The transformation is executed in a computer with a dual processor of 2.1 GHz and 4 GB of RAM memory. After the execution, transformation information is also recorded. The whole collected information is shown in Table 1.
4. After the whole execution of the sample, the collected information is statistically analyzed to answer the research questions.

4.5 Analysis Procedure

The data analysis was carried out according to the following steps:

1. The hypotheses established for RQ1 are evaluated by means of a regression line model using the Pearson linear correlation test, which quantifies the intensity of the linear relation between variables size and transformation time. Under the hypothesis that the transformation time is theoretically linear (i.e., *O(n)* with *n*=number of events), a linear regression model is established to check it and find out 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. The obtained Pearson's correlation coefficient R^2 (between -1 and 1) indicates the degree to which the real values of the dependent variable are close to the predicted values.

Table 1. Data collected in the experiment execution

ID	Size (#events)	Complexity (ECyM)	Transf. Time (s)	Performance
1	788	27	10.03	0.50
2	1564	27	41.15	0.27
3	2300	27	94.15	0.24
4	3080	27	141.62	0.13
5	1000	21	10.55	0.69
6	2000	21	41.31	0.28
7	3000	21	91.62	0.15
8	4000	21	138.11	0.11
9	628	10	76.00	1.00
10	1294	10	349.26	0.45
11	1926	10	751.34	0.26
12	2560	10	1259.48	0.22
13	646	12	93.79	0.97
14	1282	12	382.96	0.44
15	1910	12	811.55	0.27
16	2552	12	1373.64	0.23
17	972	43	22.04	0.69
18	2046	43	91.40	0.24
19	3150	43	212.45	0.16
20	3980	43	350.19	0.12
21	1200	14	33.41	0.53
22	2400	14	134.28	0.22
23	3600	14	287.99	0.13
24	4800	14	663.58	0.04
25	1842	0	23.27	0.33
26	4066	0	74.56	0.11
27	5962	0	118.06	0.05
28	7870	0	241.19	0.02
29	1094	66	21.38	0.60
30	2188	66	107.58	0.24
31	3176	66	215.36	0.16
32	4454	66	318.04	0.09
33	2408	246	27.36	0.22
34	5110	246	114.27	0.07
35	7548	246	218.29	0.01
36	9986	246	421.56	0.00
37	1904	1037	131.08	0.31
38	3800	1037	562.49	0.11
39	5934	1037	1329.71	0.04
40	7980	1037	1959.02	0.00
41	2032	0	82.63	0.28
42	4076	0	323.87	0.09
43	6116	0	763.84	0.04
44	8216	0	1512.17	0.01
45	1906	54	87.45	0.28
46	3948	54	363.11	0.10
47	6500	54	889.43	0.04
48	7668	54	1175.69	0.02
Mean	3510	127	386.32	0.24
Std. Dev.	2352	285	468.27	0.23

2. Hypotheses of RQ2 are assessed by using the ANOVA test (*Analysis Of Variance between groups*). It is a parametric test to compare how a particular factor affects the mean of a quantitative variable. If the means of variable for each factor are equal, then the factor does not affect the variable. The factor under study is the event log complexity (ECyM) labeled as low, medium and high. Each event log is categorized in these three groups according to the percentiles $Q_{1/3}$ and $Q_{2/3,}$ which divide the distribution in three sub-samples. As a result, the hypotheses of RQ2 are equivalent to the following according to the ANOVA test:

   – $H_{RQ2,0}$: $\mu_{\text{performance; ECyM=''low''}} = \mu_{\text{performance; ECyM=''medium''}} = \mu_{\text{performance; ECyM=''high''}}$. All expected means are equal.
   – $H_{RQ2,1}$: $\mu_{\text{performance; ECyM=''low''}} \neq \mu_{\text{performance; ECyM=''medium''}} \neq \mu_{\text{performance; ECyM=''high''}}$.

5 Results

The following sections show the results after analyzing the data obtained in the experiment using *R*, an open source statistical tool [27].

5.1 Scalability Testing (RQ1)

To calculate the equation of the regression line the variables were represented by a scatter plot (see Fig. 4). If the regression line is very close to most points in the scatter chart, both variables are strongly correlated. The regression line equation estimated is y = 193.23x – 291849.

After applying the Pearson correlation test, the value of linear correlation coefficient of Pearson was R^2=0.94, which is very close to 1. This value makes it possible to ensure that there is a strong positive correlation between both variables. In terms of significance, the correlation value means that 5% of the transformation time variation cannot be explained by size through the line of fit.

The result shows that the null hypothesis ($H_{RQ1,0}$) cannot be rejected since there is a linear relationship between the size of input model and transformation time.

5.2 Suitability Testing (RQ2)

The $H_{RQ2,0}$ hypothesis proposes that the means of performance for each factor of ECyM are equal. After applying the ANOVA test (see Table 2) the p-value is 0.105. Since the p-value is greater than 0.05 the null hypothesis cannot be rejected; it is accepted therefore that the means are equal at a 95% confidence level. This can also be checked graphically in Fig. 5 which shows the box chart for each distribution with low, medium and high complexity. This result proves that the complexity (ECyM) does not have an influence in the performance.

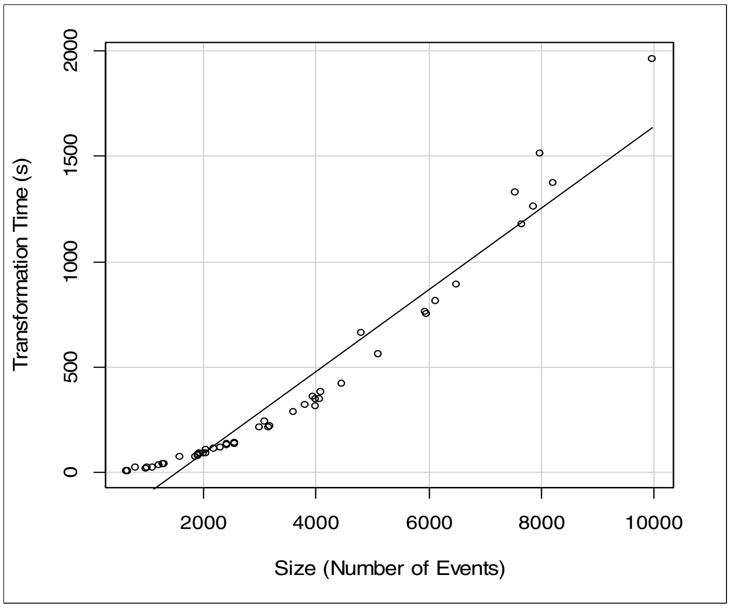

Fig. 4. Scatter plot of the size and the transformation time

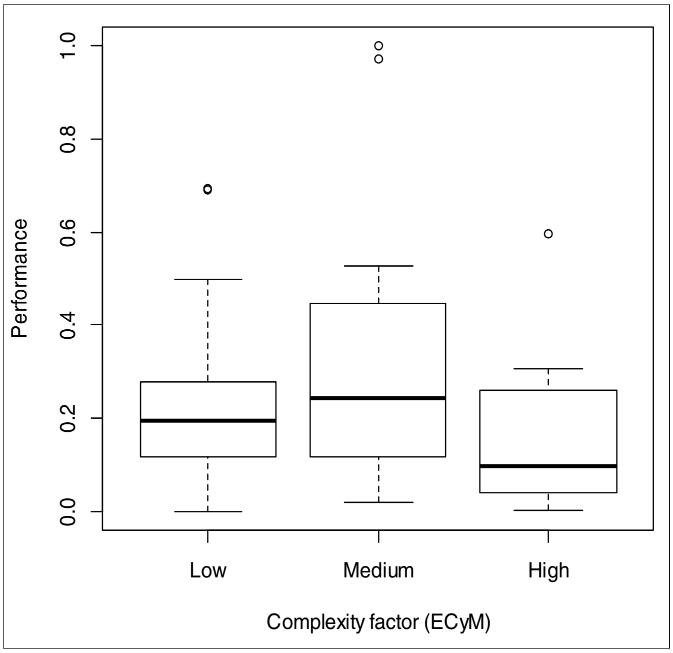

Fig. 5. Box plot of performance

Table 2. ANOVA results

	Df	Sum Sq	Mean Sq	F value	Pr (>F)
Complexity Factor	2	0.249	0.12449	2.365	**0.105**
Residuals	45	2.368	0.05263		

5.3 Validity Evaluation

This section discusses the threats to the validity of the experiment.

- **Internal Validity:** The simulation was carried out with 48 event logs simulated from randomly generated business processes. Hence, the results may differ slightly in case of generation of different business processes. In addition, the supporting tool used to obtain the business process could be a factor that may affect the values of the experiment. To mitigate this threat the experiment should be replicated by using larger samples, different tools, and then, by comparing the obtained results.
- **Construct Validity:** The selected variables were adequate to answer the research questions in an appropriate manner. However, the way in which such variables are assessed could be a threat. To mitigate this threat, other mechanisms can be considered for the evaluation of the proposed variables (e.g., complexity can be calculated using other metrics available in the literature).
- **External Validity**: The experiment considers simulated event logs, thus the obtained results could not be strictly generalized to real-life event logs. This threat may be mitigated by replicating the experiment using industrial event logs.

6 Conclusions

This paper proposes a model transformation to integrate MXML event logs into the KDM event model repository. Nowadays, KDM makes it possible to build reverse engineering tools in a KDM ecosystem where reverse engineering tools recover knowledge regarding different artifacts, and the outgoing knowledge is represented and managed in an integrated and standardized way through a KDM repository. As a result, the KDM event models can be used in combination with other embedded knowledge recovered through reverse engineering to modernize legacy information systems. This transformation therefore facilitates the applicability of business process mining techniques and algorithms within software modernization projects.

This work provides an implementation of the model transformation using QVTr as well as a supporting tool in order to facilitate its validation and adoption by the industry. In fact, the transformation is validated through an experiment based on the automatic simulation of event logs. The experiment shows that the model transformation is able to obtain KDM event models from MXML logs in a scalable and suitable way. This means that the transformation can be executed in a linear time regarding the number of events. The performance of the transformation is also independent of the complexity of the input log.

The future work will address the repeatability of the experiment using additional and different event log models in order to deal with the detected threats and to obtain strengthened conclusions.

Acknowledgments. This work was supported by the FPU Spanish Program and the R&D projects ALTAMIRA (PII2I09-0106-2463), PEGASO/MAGO (TIN2009-13718-C02-01) and MOTERO (JCCM and FEDER, PEII11-0366-9449).

References

1. Weske, M.: Business Process Management: Concepts, Languages, Architectures, Leipzig, Alemania, p. 368. Springer, Heidelberg (2007)
2. Jeston, J., Nelis, J., Davenport, T.: Business Process Management: Practical Guidelines to Successful Implementations, 2nd edn., p. 469. Butterworth-Heinemann, Elsevier Ltd., NV, USA (2008)
3. Newcomb, P.: Architecture-Driven Modernization (ADM). In: Proceedings of the 12th Working Conference on Reverse Engineering. IEEE Computer Society (2005)
4. van der Aalst, W., Weijters, A.J.M.M.: Process-aware information systems: bridging people and software through process technology. In: Dumas, M., van der Aalst, W., Ter Hofstede, A. (eds.) Process Mining, pp. 235–255. John Wiley & Sons, Inc. (2005)
5. Van der Aalst, W.M.P., et al.: ProM: the process mining toolkit. In: 7th International Conference on Business Process Management (BPM 2009) - Demonstration Track, pp. 1–4. Springer, Ulm (2009)
6. van den Heuvel, W.-J.: Aligning Modern Business Processes and Legacy Systems: A Component-Based Perspective (Cooperative Information Systems). The MIT Press (2006)
7. Pérez-Castillo, R., de Guzmán, I.G.R., Piattini, M.: Knowledge Discovery Metamodel - ISO/IEC 19506: a Standard to Modernize Legacy Systems. Computer Standards & Interfaces Journal, 519–532 (2011)
8. Pérez-Castillo, R., et al.: Integrating Event Logs into KDM Repositories. In: 27th Annual ACM Symposium on Applied Computing (SAC 2012). ACM, Riva del Garda (in Press, 2012)
9. van der Aalst, W.M.P.: Process-Aware Information Systems: Lessons to Be Learned from Process Mining. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 1–26. Springer, Heidelberg (2009)
10. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
11. Medeiros, A.K., Weijters, A.J., Aalst, W.M.: Genetic process mining: an experimental evaluation. Data Min. Knowl. Discov. 14(2), 245–304 (2007)
12. Ingvaldsen, J.E., Gulla, J.A.: Preprocessing Support for Large Scale Process Mining of SAP Transactions. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 30–41. Springer, Heidelberg (2008)
13. Günther, C.W., van der Aalst, W.M.P.: A Generic Import Framework for Process Event Logs. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 81–92. Springer, Heidelberg (2006)
14. Pérez-Castillo, R., Weber, B., García-Rodríguez de Guzmán, I., Piattini, M.: Toward Obtaining Event Logs from Legacy Code. In: Muehlen, M.z., Su, J. (eds.) BPM 2010, Part II. LNBIP, vol. 66, pp. 201–207. Springer, Heidelberg (2011)

15. Zou, Y., Hung, M.: An Approach for Extracting Workflows from E-Commerce Applications. In: Proceedings of the Fourteenth International Conference on Program Comprehension, pp. 127–136. IEEE Computer Society (2006)
16. Cai, Z., Yang, X., Wang, W.: Business Process Recovery for System Maintenance - An Empirical Approach. In: 25th International Conference on Software Maintenance (ICSM 2009), pp. 399–402. IEEE Computer Society, Edmonton (2009)
17. Di Francescomarino, C., Marchetto, A., Tonella, P.: Reverse Engineering of Business Processes exposed as Web Applications. In: 13th European Conference on Software Maintenance and Reengineering (CSMR 2009), pp. 139–148. IEEE Computer Society, Fraunhofer IESE (2009)
18. Wong, P., Gibbons, J.: On Specifying and Visualising Long-Running Empirical Studies. In: Vallecillo, A., Gray, J., Pierantonio, A. (eds.) ICMT 2008. LNCS, vol. 5063, pp. 76–90. Springer, Heidelberg (2008)
19. Syriani, E., Vangheluwe, H.: Programmed Graph Rewriting with Time for Simulation-Based Design. In: Vallecillo, A., Gray, J., Pierantonio, A. (eds.) ICMT 2008. LNCS, vol. 5063, pp. 91–106. Springer, Heidelberg (2008)
20. Biermann, E., et al.: Flexible visualization of automatic simulation based on structured graph transformation. IEEE (2008)
21. OMG, QVT. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, OMG (2008), `http://www.omg.org/spec/QVT/1.0/PDF`
22. Pérez-Castillo, R.: MXML to KDM Transformation implemented in QVT Relations (2011), `http://alarcos.esi.uclm.es/per/rpdelcastillo/modeltransformations/MXML2KDM.htm` (cited March 29, 2011)
23. Jedlitschka, A., Ciolkowski, M., Pfahl, D.: Reporting experiments in software engineering. In: Guide to Advanced Empirical Software Engineering, pp. 201–228 (2008)
24. Burattin, A., Sperduti, A.: PLG: a Framework for the Generation of Business Process Models and their Execution Logs (2011)
25. Lassen, K.B., van der Aalst, W.M.P.: Complexity metrics for Workflow nets. Information and Software Technology 51(3), 610–626 (2009)
26. ikv++, Medini QVT (2008), ikv++ technologies ag
27. R. The R Project for Statistical Computing (2011), `http://cran.r-project.org/`