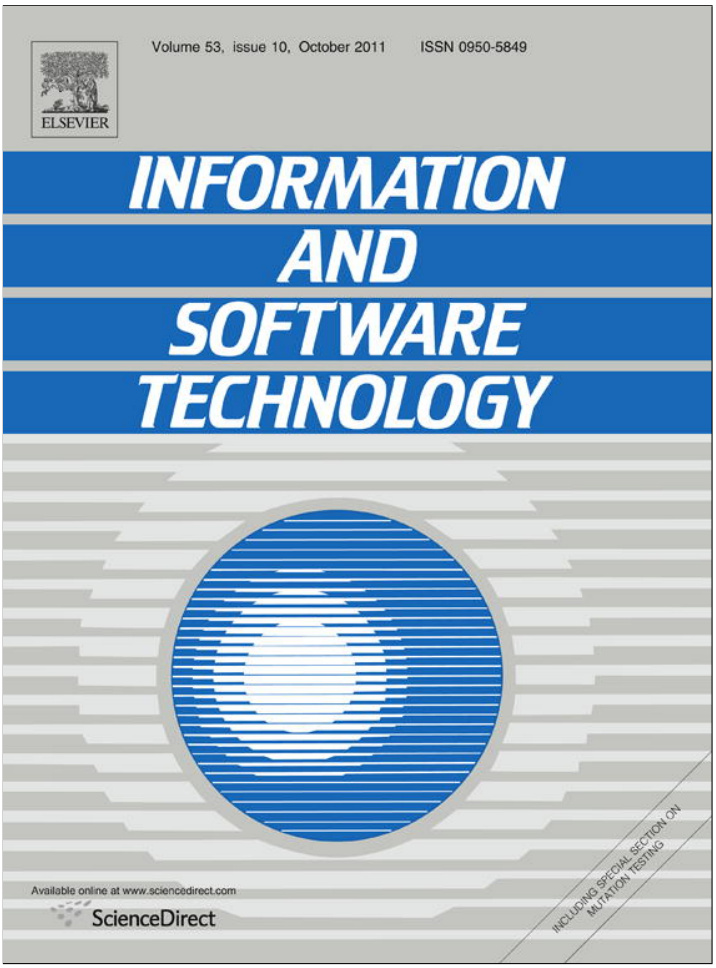


Information and Software Technology 53 (2011) 1023–1044

Contents lists available at ScienceDirect

Information and Software Technology

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

Business process archeology using MARBLE

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

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

ARTICLE INFO

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

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

ABSTRACT

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

1. Introduction

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

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

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

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

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

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

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

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

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

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

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

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

2. Background

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

2.1. Business process archeology

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

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

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

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

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

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

2.2. Architecture-Driven Modernization (ADM)

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

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

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

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

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

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

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

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

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

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

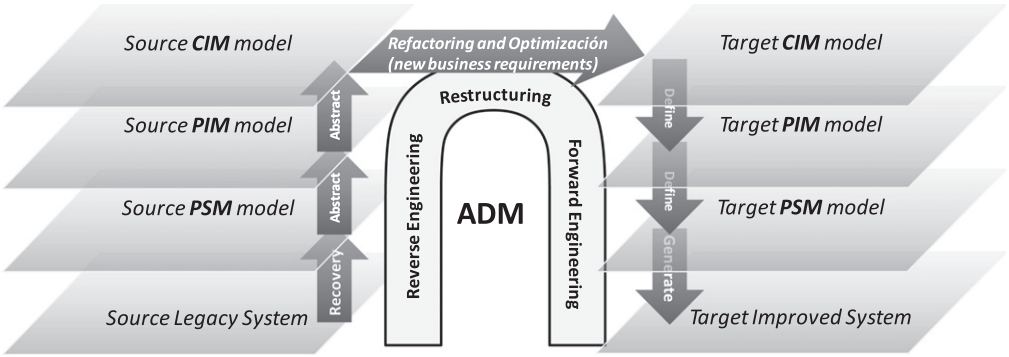


Fig. 1. Horseshoe modernization model.

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

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

3. Related work

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

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

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

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

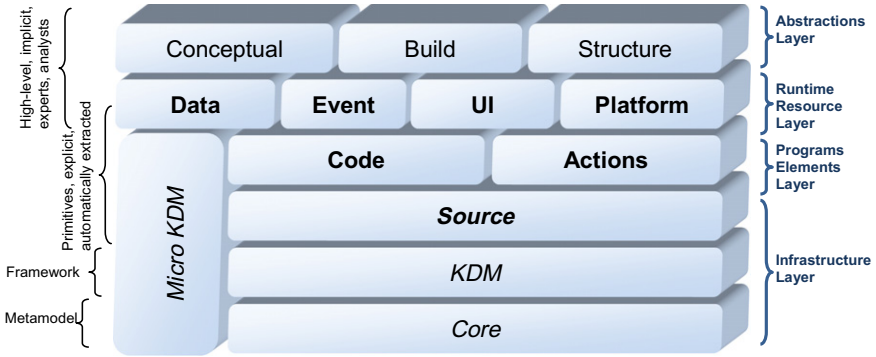


Fig. 2. Layers, packages, and concerns in KDM (adapted from [20]).

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

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

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

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

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

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

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

4. Business process archeology method

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

Table 1
Comparison between recovery processes.

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

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

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

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

4.1. MARBLE

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

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

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

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

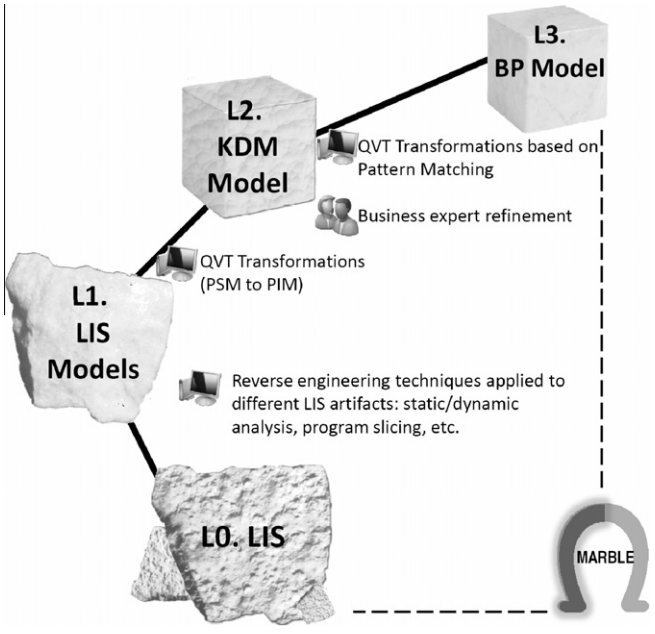


Fig. 3. MARBLE, the framework to support business process archeology.

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

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

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

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

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

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

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

4.2. L0-to-L1 transformation

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

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

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

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

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

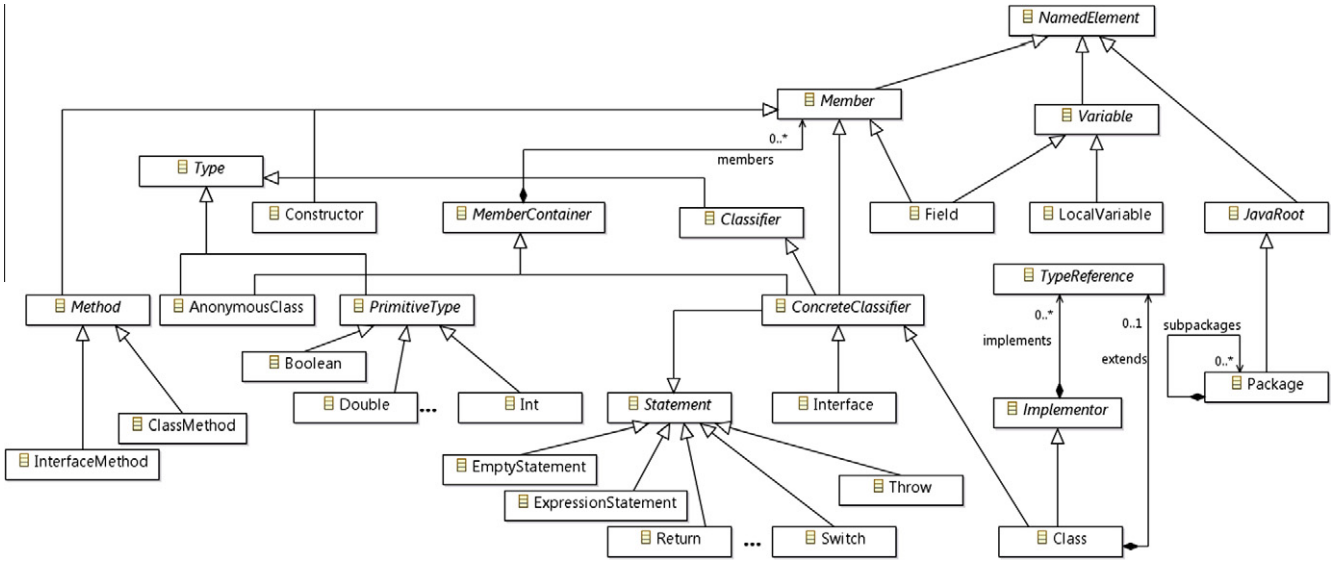


Fig. 4. A simplified view of the Java metamodel.

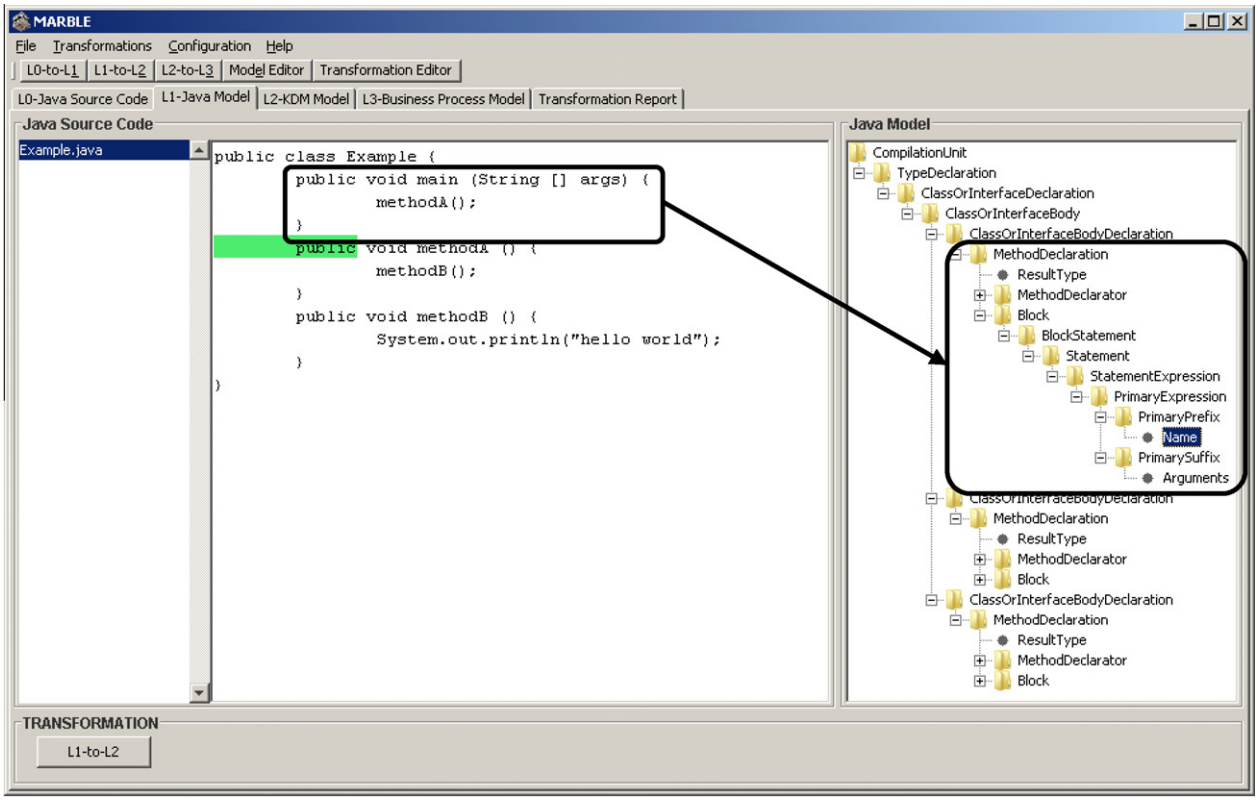


Fig. 5. A running example for a small Java file after the L0-to-L1 transformation.

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

4.3. L1-to-L2 transformation

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

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

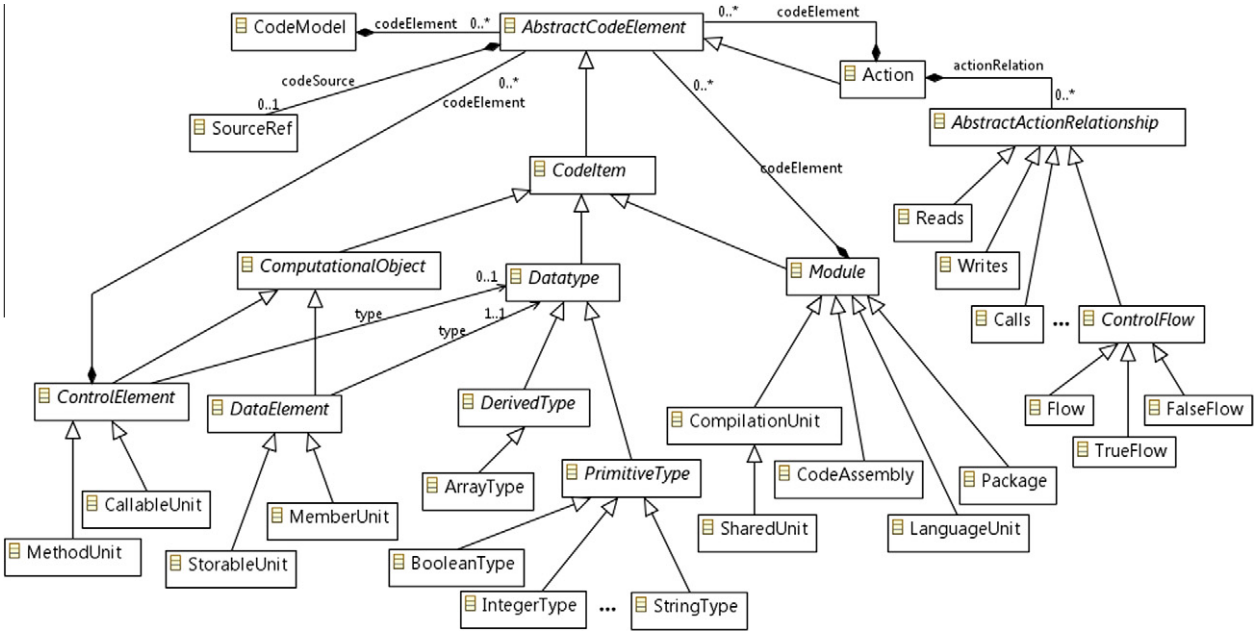


Fig. 6. A simplified view of the Code and Action packages of the KDM metamodel.

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

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

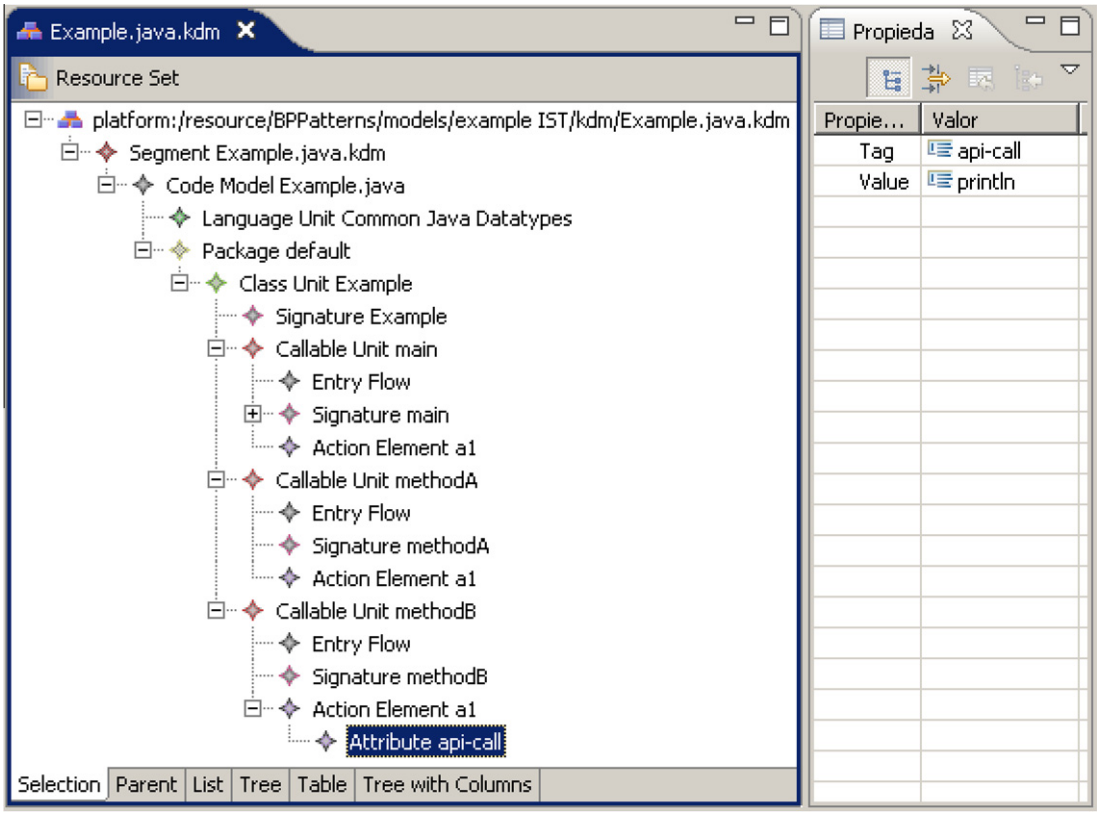


Fig. 8. An example KDM model obtained after the L1-to-L2 transformation.

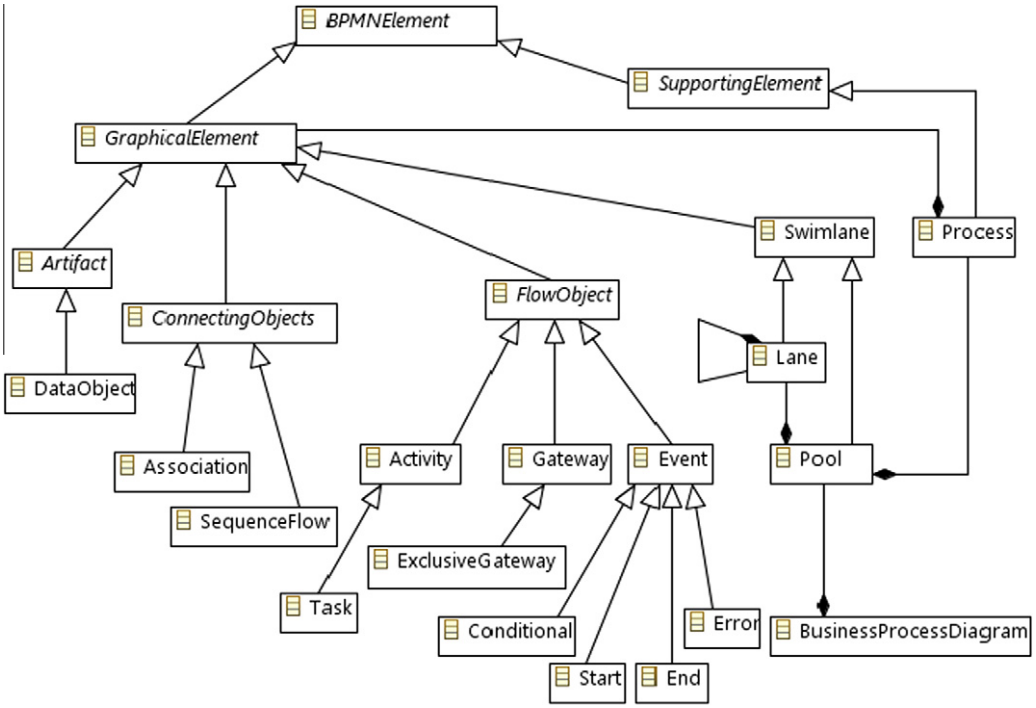


Fig. 9. BPMN metamodel.

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

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

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

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

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

4.4. L2-to-L3 transformation

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

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

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

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

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

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

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

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

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

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

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

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

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

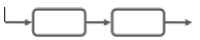
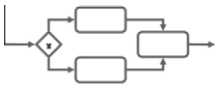
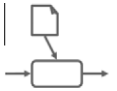
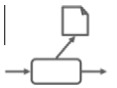
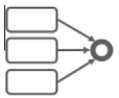
5. A tool to support business process archeology

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

5.1. Technologies involved

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

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

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

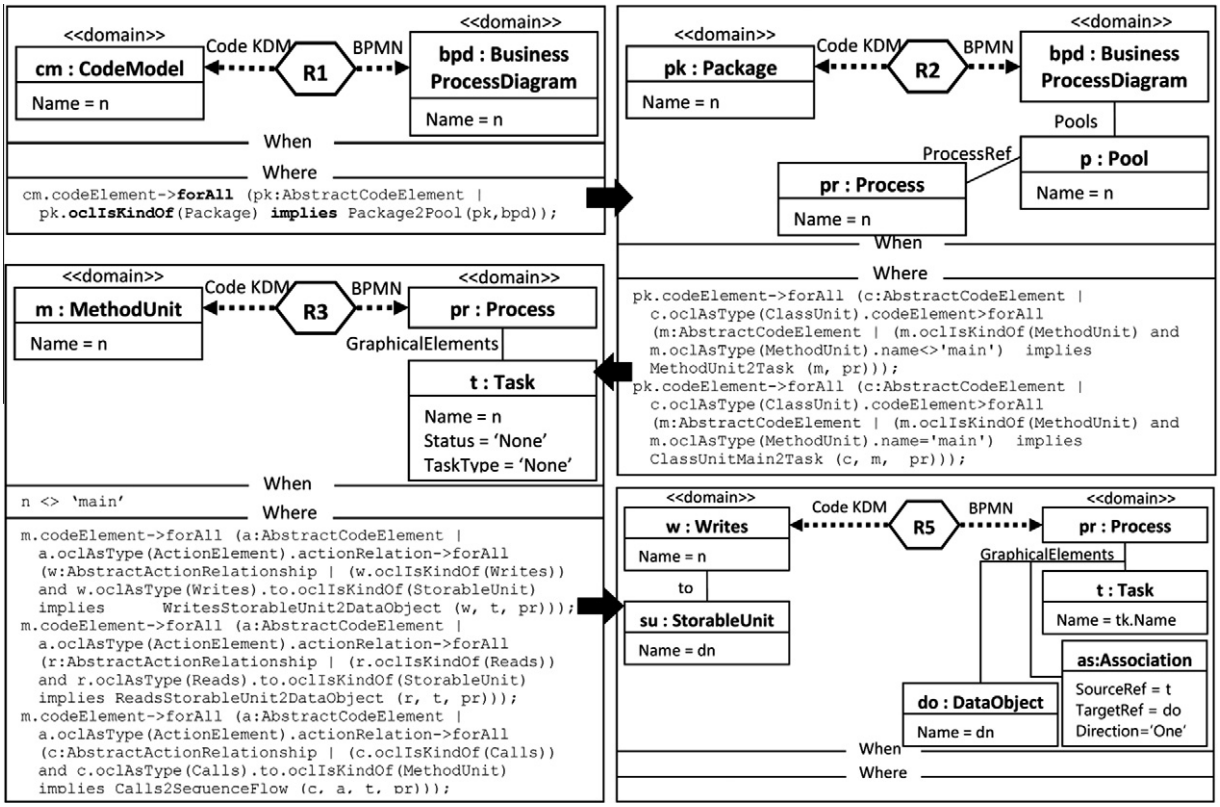


Fig. 10. Four QVT relations (graphical representation) of the L2-to-L3 transformation.

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

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

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

5.2. Tool modules

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

To support the first transformation, a tool module to carry out static analysis was developed. In this case, the module was built specifically for parsing Java source code. This tool module was developed through *JavaCC* from the *Extended Backus–Naur Form* (EBNF) grammar of Java 1.5. This module takes a Java file as input and then generates an XMI file as the output that represents the Java code model, a PSM model in L1 (see Fig. 12A).

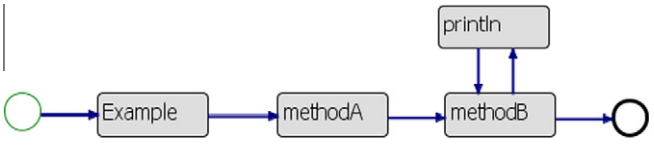


Fig. 11. An example BPMN model obtained after the L2-to-L3 transformation.

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

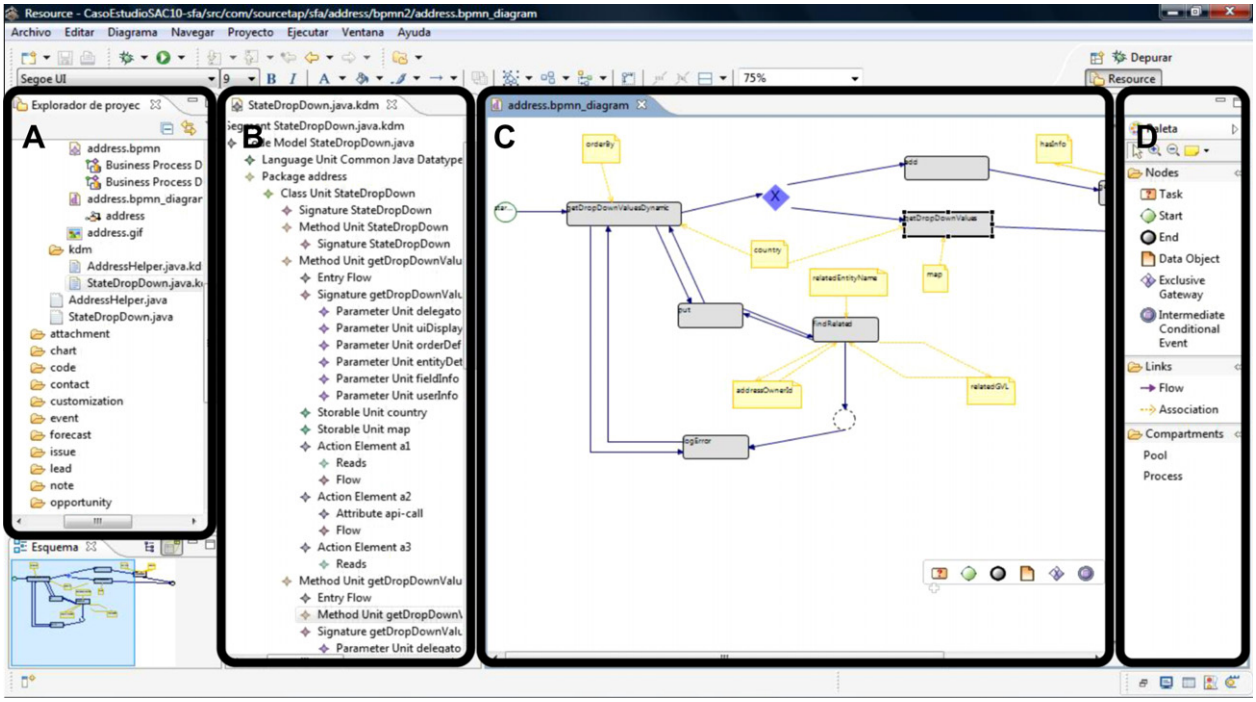


Fig. 12. The tool to support the business process archeology method.

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

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

6. Case study

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

6.1. Background

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

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

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

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

6.2. Design

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

Table 4
Case study research questions.

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

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

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

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

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

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

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

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

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

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

6.3. Case selection

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

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

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

Table 5
Criteria for case selection.

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

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

6.4. Case study procedure

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

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

6.5. Data collection

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

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

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

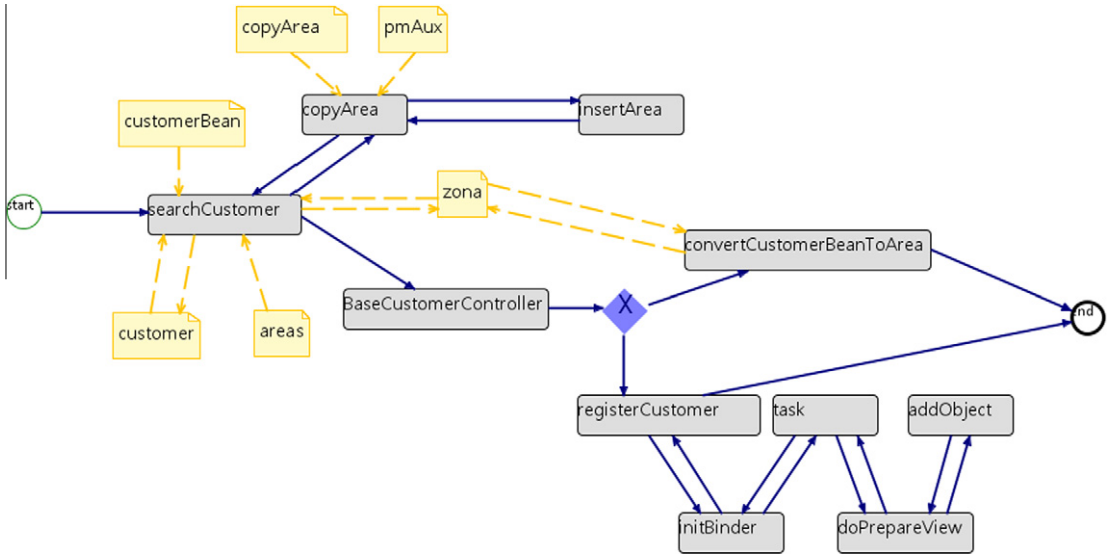


Fig. 13. An example of a BPMN model obtained in the case study.

Table 6
Data collected in the case study.

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

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

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

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

Table 7
Data from final business process models.

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

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

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

6.6. Analysis and interpretation

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

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

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

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

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

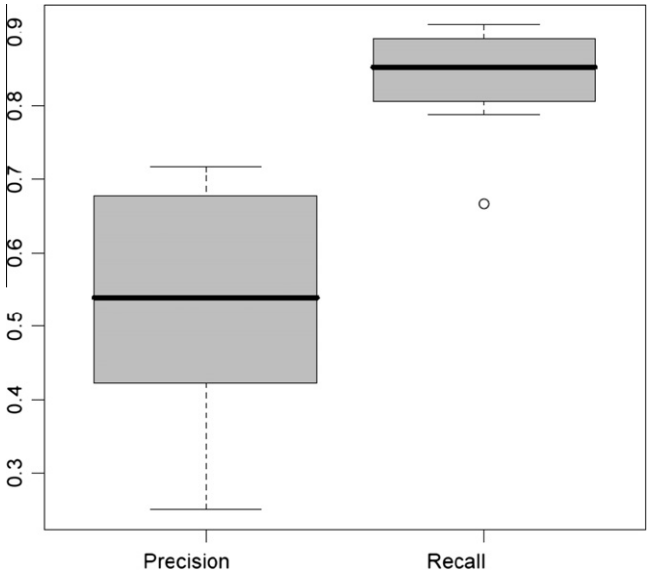


Fig. 14. The box chart for *precision* and *recall* measures.

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

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

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

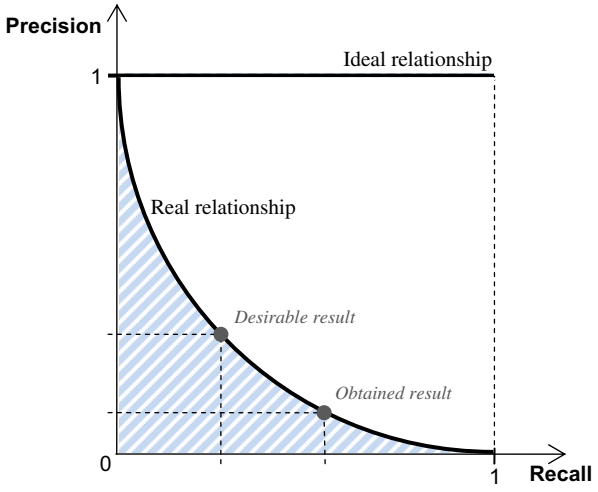


Fig. 15. Relationship between *precision* and *recall* measures and obtained results.

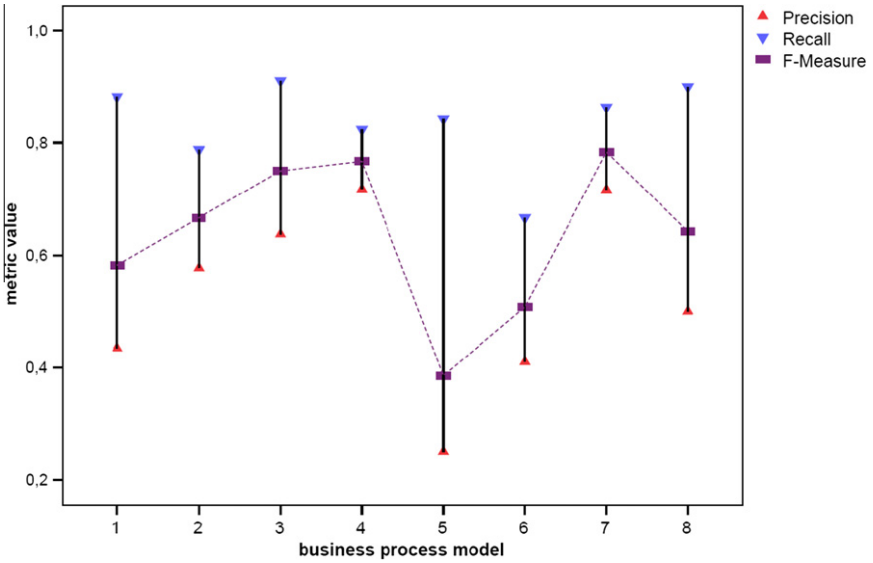


Fig. 16. F-measure values for the final business processes.

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

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

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

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

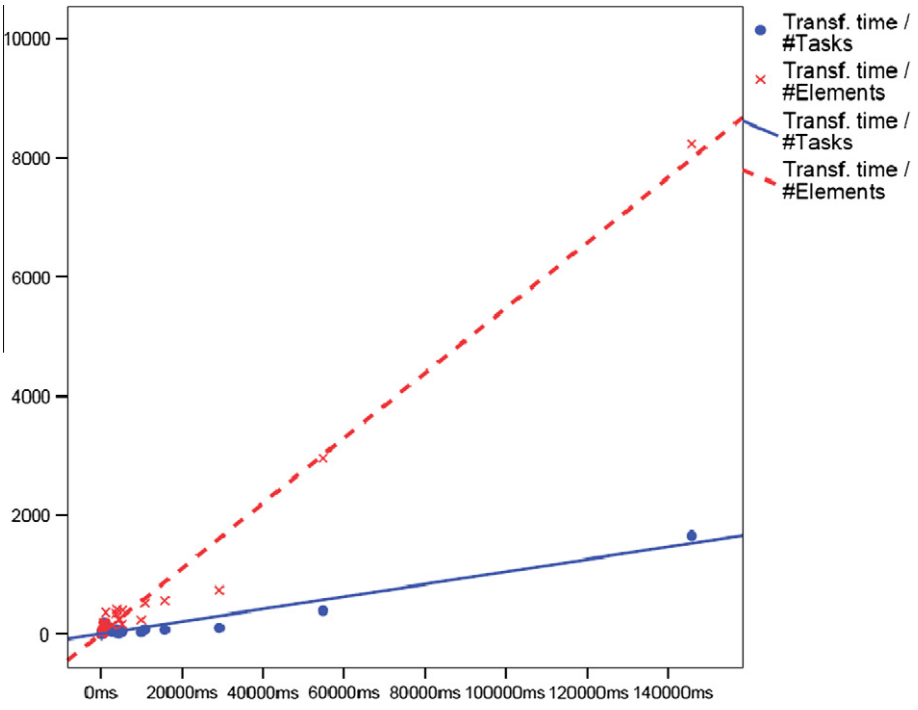


Fig. 17. The size/time scatter chart.

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

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

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

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

6.7. Validity evaluation

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

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

6.7.1. Internal validity

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

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

6.7.2. Construct validity

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

6.7.3. External validity

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

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

7. Conclusions and future work

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

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

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

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

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

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

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

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

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

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

Acknowledgments

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

References

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

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