SOFTWARE – PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2012; **42**:159–189
Published online 11 February 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.1057

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

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

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

SUMMARY

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

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

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

1. INTRODUCTION

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

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

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

Copyright © 2011 John Wiley & Sons, Ltd.

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

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

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

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

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

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

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

This paper presents an entire functional business process recovery procedure framed in MARBLE, which is specially implemented for object-oriented systems. The proposed procedure is aligned with the four different abstraction levels proposed in MARBLE: L0 to represent the legacy system; L1, which contains different platform-specific models (PSM) to depict different software artifacts of the legacy system; L2, which integrates all the specific L1 models into a platform-independent model (PIM), which is represented according to KDM (Knowledge Discovery Metamodel) [16]; and finally L3 to represent the business process model.

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

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

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

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

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

2. BACKGROUND

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

2.1. ADM

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

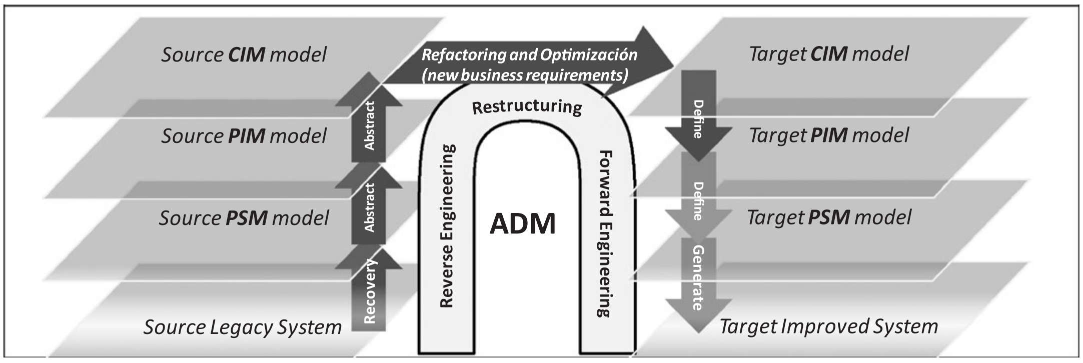


Figure 1. Horseshoe modernization model.

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

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

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

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

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

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

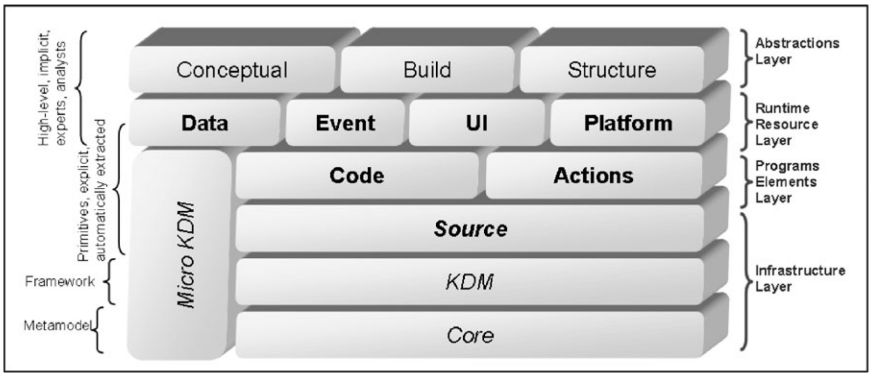


Figure 2. Layers, packages, and concerns in KDM (adapted from [16]).

2.2. KDM

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

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

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

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

3. RELATED WORK

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

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

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

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

Table I. Comparison between business knowledge recovery proposals.

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

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

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

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

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

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

4. BUSINESS PROCESS RECOVERY PROCEDURE

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

4.1. MARBLE

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

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

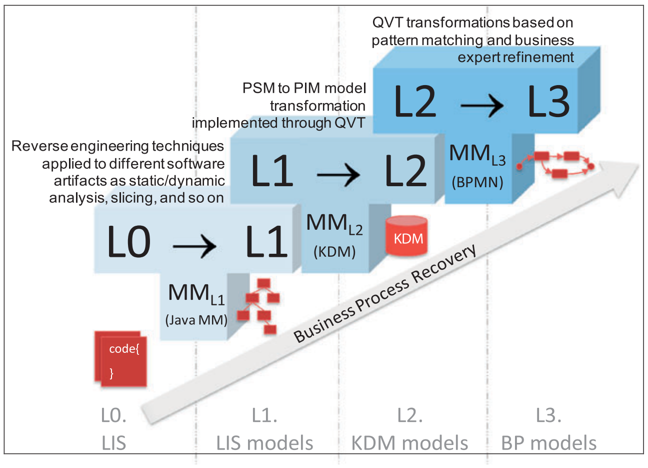

QVT transformations based on pattern matching and business expert refinement
L2 → L3
PSM to PIM model transformation implemented through QVT
L1 → L2
Reverse engineering techniques applied to different software artifacts as static/dynamic analysis, slicing, and so on
L0 → L1
MM_L3 (BPMN)
MM_L2 (KDM)
KDM
MM_L1 (Java MM)
code{ }
Business Process Recovery
L0. LIS
L1. LIS models
L2. KDM models
L3. BP models

Figure 3. Overview of MARBLE.

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

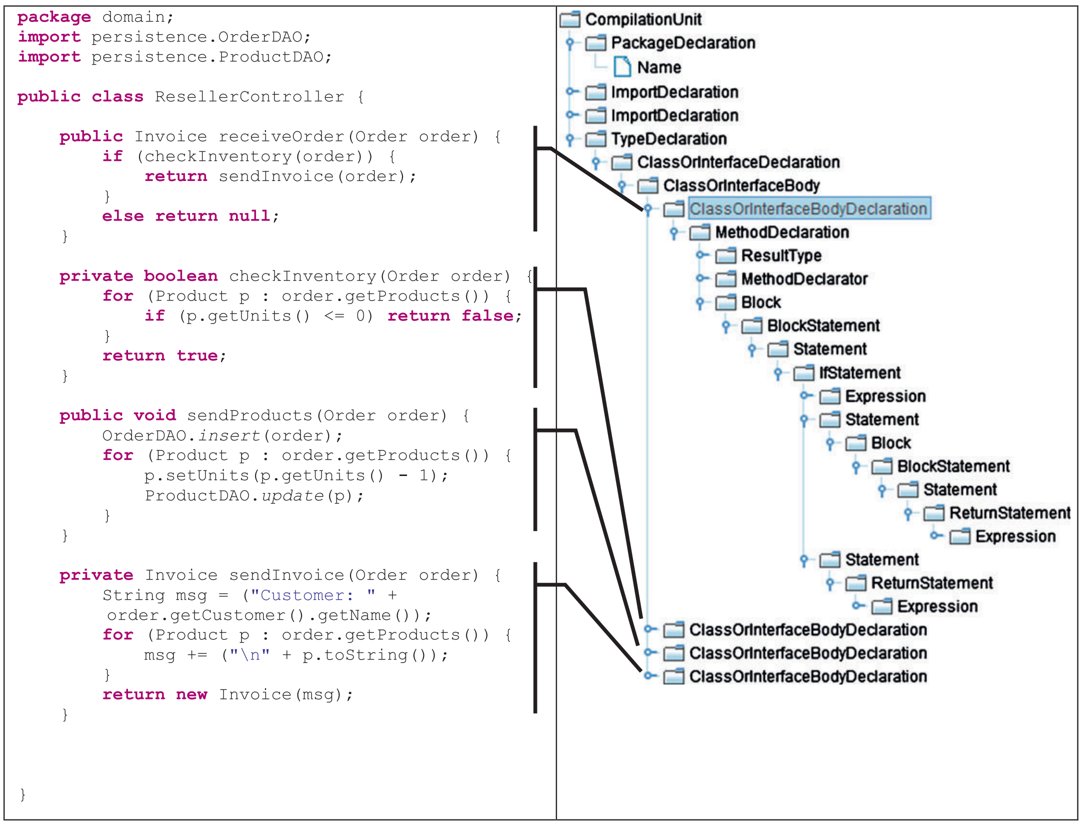
The proposed business process recovery procedure defines the three concrete transformations between these four levels. These transformations are presented in the following subsections.

4.2. L0-to-L1 transformation

The first transformation takes the different software artifacts from legacy information systems and obtains a specific model for each one. The software artifacts considered in this transformation depend on the specific business process recovery procedure framed in MARBLE. If the specific method considers more artifacts, it will probably have more sources from which to extract the business knowledge needed.

We are currently contemplating a recovery procedure framed in MARBLE, which considers legacy source code as the unique software artifact since, according to [40], it is the artifact that embeds most business knowledge. The L0-to-L1 transformation thus obtains a code model that represents the source code of the legacy system at a low abstraction level, i.e. it considers technological details. Indeed, this transformation is, in this case, tuned to analyze Java-based systems. Nevertheless, this transformation can be extended in each case by considering more program languages, which will imply more than one code model at L1, although those code models would be then integrated at L2 through the KDM model.

The analysis can be carried out by means of different reverse engineering techniques such as static analysis which examines the source code files, dynamic analysis which analyzes the source code at runtime, and program slicing which studies the code and divides it into different fragments. This transformation uses static analysis as the reverse engineering technique to extract the necessary information. The transformation is therefore supported by a parser that syntactically analyzes the Java source files of the legacy information system from a static viewpoint and builds Java code models according to a Java metamodel. In particular, the parser analyzes the text of a Java source file, considering it as a sequence of tokens (e.g. keywords, identifiers, and so on) to



```
package domain;
import persistence.OrderDAO;
import persistence.ProductDAO;

public class ResellerController {

    public Invoice receiveOrder(Order order) {
        if (checkInventory(order)) {
            return sendInvoice(order);
        }
        else return null;
    }

    private boolean checkInventory(Order order) {
        for (Product p : order.getProducts()) {
            if (p.getUnits() <= 0) return false;
        }
        return true;
    }

    public void sendProducts(Order order) {
        OrderDAO.insert(order);
        for (Product p : order.getProducts()) {
            p.setUnits(p.getUnits() - 1);
            ProductDAO.update(p);
        }
    }

    private Invoice sendInvoice(Order order) {
        String msg = ("Customer: " +
         order.getCustomer().getName());
        for (Product p : order.getProducts()) {
            msg += ("\n" + p.toString());
        }
        return new Invoice(msg);
    }


}
```

```
CompilationUnit
  PackageDeclaration
    Name
  ImportDeclaration
  ImportDeclaration
  TypeDeclaration
    ClassOrInterfaceDeclaration
      ClassOrInterfaceBody
        ClassOrInterfaceBodyDeclaration
          MethodDeclaration
            ResultType
            MethodDeclarator
            Block
              BlockStatement
                Statement
                  IfStatement
                    Expression
                    Statement
                      Block
                        BlockStatement
                          Statement
                            ReturnStatement
                              Expression
                    Statement
                      ReturnStatement
                        Expression
        ClassOrInterfaceBodyDeclaration
        ClassOrInterfaceBodyDeclaration
        ClassOrInterfaceBodyDeclaration
```

Figure 4. An example of the L0-to-L1 transformation for a product shipping system.

determine whether the text represents a valid Java code file. The parser builds an abstract syntax tree of the Java code file on the fly, which represents a Java code model at L1.

An example will now be shown to illustrate the performance of the proposed transformations. Let us imagine an information system supporting the product shipping of a reseller organization. This example focuses on a certain piece of source code: the *ResellerControler* class of the *domain* package (see Figure 4, left side). This class contains four methods that support four key functionalities of the system: (i) *receiveOrder* manages customers' order requests; (ii) *checkInventory* checks whether the products needed to fulfill a specific order are in stock; (iii) *sendProducts* manages the dispatch of the products ordered; and finally (iv) *sendInvoice* generates and sends the invoice to the customer.

According to the example, the L0-to-L1 transformation takes the Java source file (see Figure 4, left-hand side) and obtains the abstract syntax tree representing the Java code model at L1 (see Figure 4, right-hand side). The parser obtains a *CompilationUnit* element for each of the Java source files analyzed, and then adds the *PackageDeclaration* and *ImportDeclaration* elements. The parser then generates a *ClassOrInterfaceDeclaration* element for the class, which contains a *ClassOrInterfaceBodyDeclaration* element for each method. Each method is represented through a *MethodDeclaration* with a *ResultType*, *MethodDeclarator*, and a *Block* of *Statement* elements. The *Statement* element is, in turn, specialized into several kinds of elements: *ReturnStatement*, *IfStatement*, *Expression*, and so on. The right-hand side of Figure 4 shows the elements used to represent the business logic implemented in the *receiveOrder* method.

4.3. *L1-to-L2 transformation*

The L1-to-L2 transformation is in charge of the transformation of the PSM models of L1 into a single PIM model in L2 according to the KDM metamodel. The KDM model in L2 considers the

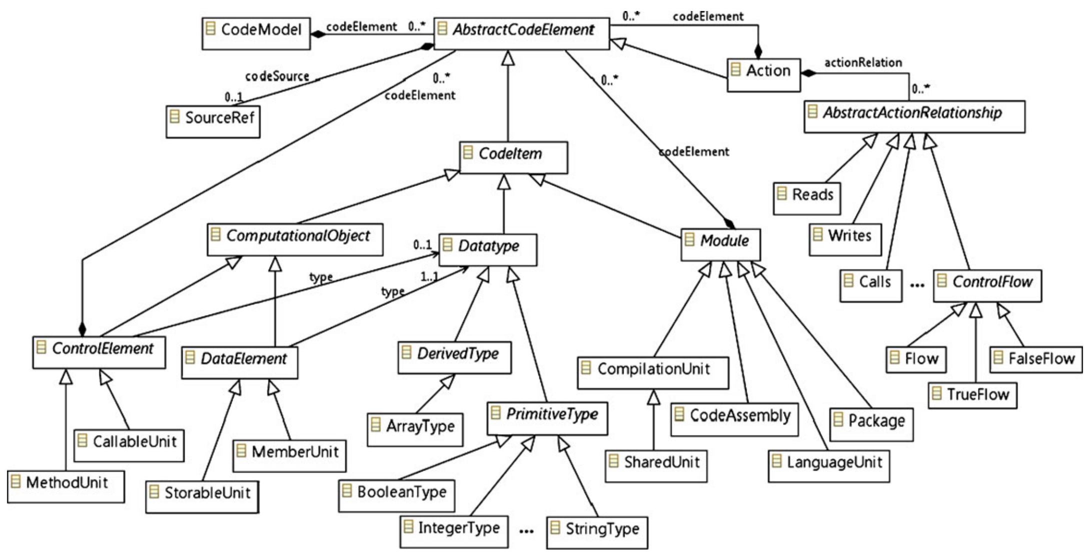


Figure 5. Overview of action and code packages of the KDM metamodel.

code and *action* KDM packages that conform to the *program element* layer of the KDM metamodel (see Figure 2). The proposal uses only these packages because legacy source code is the unique artifact considered at L0 according to our procedure design.

Figure 5 shows the most important metaclasses in the *code* and *action* packages of the KDM metamodel. According to the KDM *code* metamodel, each of the legacy systems analyzed must be represented as an instance of the *CodeModel* metaclass, which is the root metaclass. A *CodeModel* is composed of *AbstractCodeElements*, a metaclass that represents an abstract parent class for all KDM entities that can be used as *CallableUnit*, *StorableUnit*, and so on. The *CodeElement* metaclasses are also interrelated by means of *AbstractCodeRelationships* (*action* package),a metaclass representing an abstract parent for all KDM relationships that can be used to represent the code relationships, such as *Flow*, *Calls*, *Reads*, and *Writes*.

In order to establish the model transformation between the code model (the PSM model at L1) and the KDM code model (the PIM model at L2), we use QVT to define the transformation. The L1-to-L2 transformation specifically uses the QVT-Relation language, since the implementation of the transformation is easy in a declarative manner. This is owing to the fact that the structures of the Java metamodel (L1) and KDM code-action metamodel (L2) are very similar. Indeed, in some cases the transformation is as simple as renaming the metaclasses between the two metamodels, although the transformation may not be as trivial for other elements.

A QVT transformation consists of several relations focusing on the transformation of specific elements. Each relation defines at least two domains of elements: (i) the source domain, tagged as *checkonly*, which evaluates whether the specific configuration of elements of the input metamodel (Java metamodel) exists, and the target domain, tagged as *enforce*, which evaluates the configuration of elements of the output metamodel (KDM) and creates or destroys elements to satisfy the rules of the relation. A QVT relation can also have *when* or *where* clauses which establish the pre- and post-conditions of execution. Figure 6 shows a fragment of the proposed QVT transformation: the *class2compilationUnit* and *method2callableUnit* QVT relations. The *class2compilationUnit* relation transforms all the instances of the *Class* metaclass in the Java code model (L1) into instances of *CompilationUnit* in the KDM model (L2). The relation also examines the *Method* instances belonging to the *Class* instance in the where clause (see Figure 6) and the *method2callableUnit* relation is triggered. The *method2callableUnit* relation is in charge of the transformation of the instances of the *Method* metaclass into *CallableUnit* instances with the same name and type. All the different kinds of statements in the Java code model are then transformed into *codeElement* instances by means of various QVT relations triggered in the where clause.

```
transformation Java2KDM (java:java, kdm:code){
  ...
  relation class2compilationUnit {
     className : String;
     checkonly domain java jc : java::classifiers::Class {
        name = className
     };
     enforce domain kdm kp : KDM_MetaModel::code::Package {
        codeElement = kcu : KDM_MetaModel::code::CompilationUnit {
           name = className
        }
     };
     where {
        jc.members->forAll (jm:java::members::Method | jm.oclIsKindOf(java::members::Method)
           implies method2callableUnit (jm, kcu));
     }
  }
  relation method2cllableUnit {
     methodName : String;
     methodType : String;
     checkonly domain java jm : java::members::Method {
        name = methodName,
        typeReference = jtr : java::types::TypeReference {
           name = methodType
        }
     };
     enforce domain kdm kcu : KDM_MetaModel::code::CompilationUnit {
        codeElement = cu : KDM_MetaModel::code::CallableUnit {
           name = methodName,
           type = t : KDM_MetaModel::code::Datatype {
              name = methodType
           }
        }
     };
     where {
        jm.parameters->forAll(jp:java::parameters::Parameter | jp.oclIsKindOf(java::parameters::
           Parameter) implies parameter2codeElement(jp, cu));
        jm.members->forAll(js:java::statements::Statement | js.oclIsKindOf(java::statements::
           ExpressionStatement) implies expressionStatement2codeElement(js, cu));
        jm.members->forAll(js:java::statements::Statement | js.oclIsKindOf(java::statements::
           Conditional) implies conditional2codeElement(js, cu));
        ...
        jm.members->forAll(js:java::statements::Statement | js.oclIsKindOf(java::statements::
           Return) implies return2codeElement(js, cu));
     }
  }
  ...
}
```

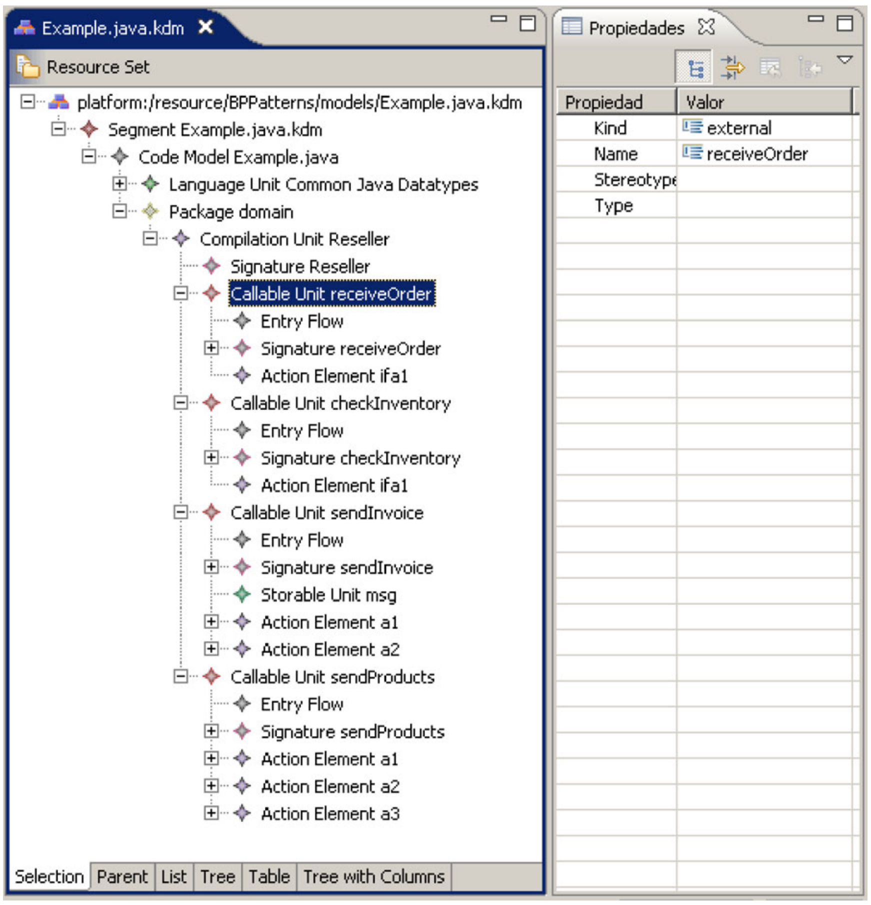
Figure 6. QVT relations to transform Java code models into KDM models.

To continue with the example, Figure 7 shows the KDM model obtained after the execution of the proposed QVT transformation for the product shipping system. The KDM model contains a *Package* instance named *domain* with a nested *CompilationUnit* instance named *Reseller*. The *CompilationUnit* is obtained from the *Class* instance of the Java code model at L1. The *Reseller CompilationUnit* instance contains an instance of the *CallableUnit* metaclass for each Java method at L1. Each *CallableUnit* instance is also defined by means of different *CodeElement* and *ActionElement* instances. For example, the *receiveOrder* method is modeled as a *CallableUnit* containing (see Figure 7): (i) an *EntryFlow* instance that defines the first KDM action in the unit (the first Java statement, since a model's sequentiality of actions is not clearly defined); (ii) a *Signature* instance that defines the parameters of the unit; and finally (iii) an *ActionElement* instance that represents the *if* statement in the Java method. The remaining *CallableUnit* instances have a similar structure.

4.4. L2-to-L3 transformation

The L2-to-L3 transformation is the last transformation and consists of two steps: (i) a model transformation that obtains a first version of business process models and (ii) manual post-intervention by business experts who modify and refine the business processes obtained in order to improve them.

First, the model transformation establishes a set of business patterns [41], which define which pieces of the source code (represented in a KDM code model) are transformed into specific



```
platform:/resource/BPPatterns/models/Example.java.kdm
  Segment Example.java.kdm
    Code Model Example.java
      Language Unit Common Java Datatypes
      Package domain
        Compilation Unit Reseller
          Signature Reseller
          Callable Unit receiveOrder
            Entry Flow
            Signature receiveOrder
            Action Element ifa1
          Callable Unit checkInventory
            Entry Flow
            Signature checkInventory
            Action Element ifa1
          Callable Unit sendInvoice
            Entry Flow
            Signature sendInvoice
            Storable Unit msg
            Action Element a1
            Action Element a2
          Callable Unit sendProducts
            Entry Flow
            Signature sendProducts
            Action Element a1
            Action Element a2
            Action Element a3
```

Propiedad	Valor
Kind	external
Name	receiveOrder
Stereotype	
Type	

Figure 7. An example of the L1-to-L2 transformation for a product shipping system.

structures of a business process. The business process models at L3 are represented according to the BPMN metamodel (see Figure 8). The BPMN metamodel represents *business process diagrams* (BPD) that involve four kinds of elements: (i) flow object elements such as *events*, *activities*, and *gateways*; (ii) connecting object elements, such as *sequence flows*, *message flows*, and *associations*; (iii) artifact elements, such as *data objects*, *groups*, and *annotations*; and (iv) swim lane elements for grouping elements, such as *pools* and *lanes*.

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

Table II presents the set of proposed business patterns of the L2-to-L3 transformation. The patterns take well-defined business patterns from the literature that have been successfully applied by business experts in the modeling of business processes [41–44]. These business patterns are adapted in order to consider which specific structure of the legacy system (represented in a KDM model) is transformed into each specific business pattern. Each pattern therefore consists of (i) a source configuration or structure of elements in the input model (KDM model in L2) and (ii) a target structure of elements in the output model (BPMN model in L3).

In order to support the set of patterns, a model transformation is implemented using QVT-Relations [17], which supports the pattern matching process. Each QVT relation searches for instances of the source structures defined by each pattern. The QVT relations, which are defined in a declarative manner, then enforce the creation in the business process model of instances of the target structures of the pattern for each input instance found in the KDM model.

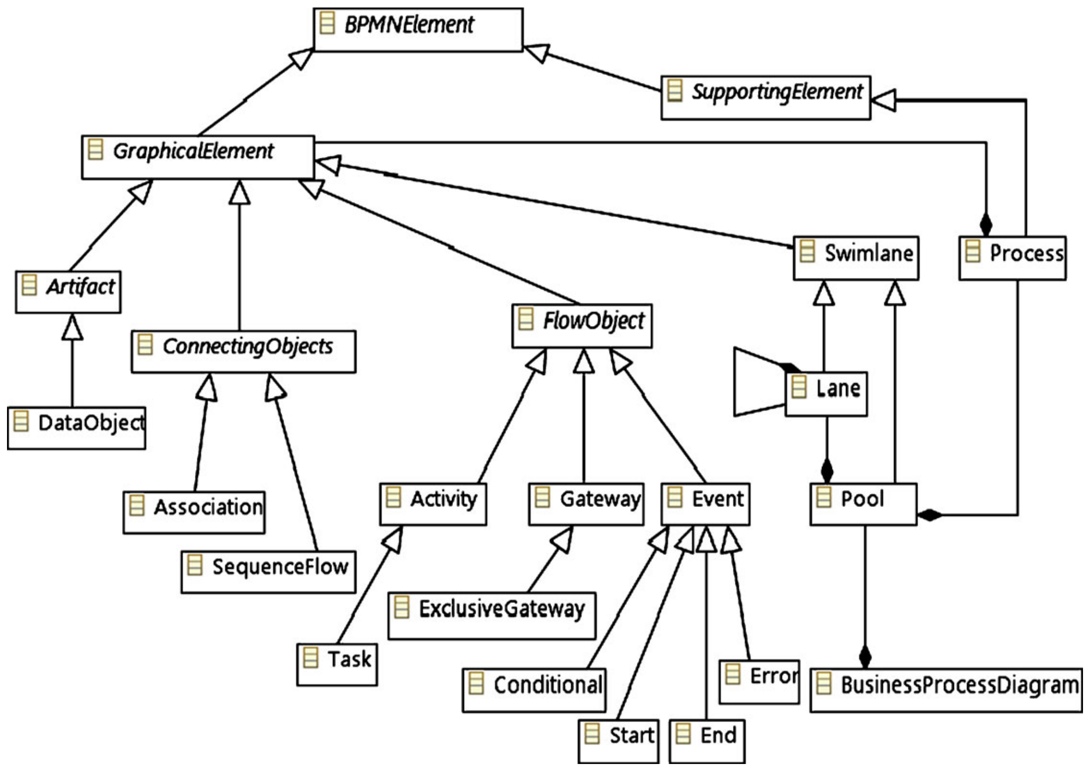
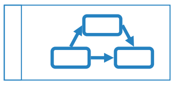
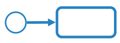
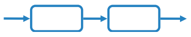
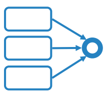
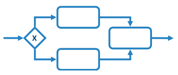
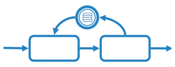
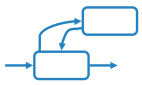
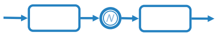
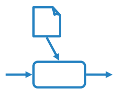
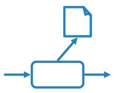


Figure 8. BPMN metamodel.

Figure 9 shows a fragment of the QVT transformation: the *Package2Pool* and *CallableUnit2Task* relations. First, the *Package2Pool* relation transforms each instance of the *Package* metaclass into a *Process* instance nested in a *Pool* instance with the same name. This transformation builds the BPD skeleton in the business process model at L3 according to the $P1$ pattern (see Table II). The *where* clause additionally triggers the *CallableUnit2Task* relation for each *CallableUnit* nested in a *CompilationUnit* that belongs to the Package instance. Second, the *CallableUnit2Task* relation transforms the *CallableUnit* instances into *Task* instances within the BPD supporting the $P2$ pattern (see Table II). In the where clause, this relation calls the *WritesStorableUnit-2DataObject* relation from among other relations, which implements the $P6$ pattern, that is called for each piece of data written by the callable unit.

After the QVT transformation, business experts can modify the recovered business process models in order to tune the organization's processes. On the one hand, this is owing to the fact that the business processes obtained may contain certain elements related to the technical nature of a legacy system, which do not depict any of the organization's business behavior. These elements can therefore be removed, or perhaps renamed. On the other hand, the preliminary business processes may lack certain business elements that depict specific aspects of the organization's business operation. They should thus be added to the recovered business processes. In this last transformation the procedure therefore defines a set of manual operations that the business experts can carry out. The finite set of operations consists of *Add* [A], *Remove* [R], *Rename* [RN], *Join* [J], and *Split* [S]. The *Join* and *Split* operations can be applied solely to whole business processes or activities, while the remaining operations can be applied to any kind of business element. In addition, the *Join* and *Split* operations can be used together with the addition of *Lanes* to represent information about several subsystems. Lanes representing subprocesses which are related by means of certain data objects written by a subprocess and read by another one (e.g. data stored in a database by a subsystem and loaded by other subsystem; or a subsystem invoking a service of another one).

Table II. Summary of the set of business patterns.

Sequence patterns	*Event patterns*
 *P1. BPD skeleton.* This pattern creates the organizational structure of a business process model. It creates a business process diagram for each KDM code model and a pool element with a nested process in the diagram for each package of the KDM code model.	 *P5. Start.* The task building from the callable unit that starts the execution of any program or application of the legacy system is considered as the initial task. A start event is therefore built into the BP diagram and a sequence flow from this event to the initial task is also created.
 *P2. Sequence.* This pattern takes any callable pieces of code from the KDM code model and maps them into tasks in the BP diagram. The sequence of calls to callable units is transformed into a set of sequence flows in the tasks built from the callable unit.	 *P6. Implicit termination.* This pattern builds an end event in the BP model. It then creates sequence flows from 'end task' and these flows merge in the end event. A task is considered to be an 'end task' if this task does not have any outgoing sequence flow.
 *P3. Branching.* This pattern transforms each conditional jump of the source code that has two mutually exclusive choices into an exclusive gateway and two different sequence flows in the business process model. These exclusive conditional branches are typically related to the *if-then-else* or *switch* statements in several programming languages.	 *P7. Conditional sequence.* This pattern transforms each conditional call into a sequence flow triggered under a conditional intermediate event through to the task related to the callable unit. It makes it possible to create arbitrary cycles in the BP diagram.
 *P4. Collaboration.* Each call to external callable unit (i.e. API libraries or external components outside the system like remote callable unit) is transformed into both an auxiliary task and two sequence flows in a round-trip manner.	 *P8. Exception.* Each call to callable unit that manages any exception is transformed into a task and a sequence flow fired under an error intermediate event. It can be considered as a specialization of the P7 pattern.
Data patterns	
 *P9. Data input.* This pattern transforms each piece of input data within a callable unit in the KDM code model into a data object and an association between the data object and the task previously built from the callable unit. It only considers as input data the parameters or arguments of the callable unit, but it does not consider the auxiliary variables within the callable unit.	 *P10. Data output.* Each piece of output data involved in a callable unit is transformed into a data object and an association from the task (built from the callable unit) to the data object. It excludes as output data the auxiliary and intermediate data in the body of the callable unit. The output data is the data returned by the callable unit or external data related to databases or files.

In order to reduce the subjective viewpoint of the business experts, some recommendations are provided to the business experts. These suggestions aid the decision-making process carried out by business experts. The recommendations are automatically defined by analyzing and scoring certain elements of the obtained business process models. There are mainly two kinds of recommendations. The first kind of recommendations is provided when a business process (obtained from a particular code package) has several business tasks related to other business processes (obtained from other code package by applying the pattern *P4. Collaboration*), it suggests joining both business process models obtained from two different code packages. The second kind of recommendation is provided whether a certain business process model consists of two (or more) isolated connected graphs, it suggests then that business experts split the model into two (or more) business process models.

To continue with the example, at the beginning the L2-to-L3 transformation obtains a first sketch of the BPD from the KDM model by means of the QVT transformation. Figure 10(A) shows the graphical representation of the business process model obtained during the QVT transformation.

```
transformation patterns (kdm:code, bpmn:bpmn){
   ...
   relation Package2Pool {
      xName : String;
      checkonly domain kdm pk : code::Package  {
         name = xName
      };

      enforce domain bpmn bpd: bpmn::BusinessProcessDiagram  {
         Pools = p : Pool {
            name = xName,
            ProcessRef = pr : bpmn::Process {
               Name = xName
            }
         }
      };
      where {
         pk.codeElement->forAll (c:code::AbstractCodeElement |
         c.oclAsType(code::CompilationUnit).codeElement->forAll(m:code::AbstractCodeElement |
         (m.oclIsKindOf(code:: CallableUnit) and m.oclIsUndefined() and m.oclAsType(code::
         CallableUnit).name<>'main')  implies   CallableUnit2Task (m, pr)));
         ...
      }
   }
   ...
   relation CallableUnit2Task {
      xName : String;
      checkonly domain kdm m : code::CallableUnit {
         name = xName
      };
      enforce domain bpmn pr : bpmn::Process {
         GraphicalElements = t : bpmn::Task {
            Name = xName,
            Status = bpmn::StatusType::None,
         }
      };
      where {
         m.codeElement->forAll (a: AbstractCodeElement | a.oclAsType
         (ActionElement).actionRelation->forAll (w:AbstractActionRelationship |
         (w.oclIsKindOf(Writes)) and w.oclAsType(Writes).to.oclIsKindOf
         (StorableUnit) implies WritesStorableUnit2DataObject (w, t, pr)));
         ...
      }
   }
}
```

Figure 9. QVT relations to implement the L2-to-L3 transformation.

This model contains 10 tasks in total, although only four tasks are related to the four *CallableUnit* instances in the KDM model (see Figure 7), which are obtained by applying the *P2.Sequence* pattern. The gateways are also created by applying the *P3. Branching* pattern. Moreover, six other tasks are also obtained by applying the *P4.Collaboration* pattern (see Table II). This pattern is applied to the three *ActionElement* instances of the *CallableUnit* instance named *sendProducts* in the KDM model (see Figure 7). These action elements represent calls to methods not defined in the same Java source file (e.g. *update*, *setUnits*, and *insert*), and these calls are thus represented as external calls, which are transformed into six tasks with a round-trip sequence flow from the previous tasks.

Figure 10(B) shows the business process model refined by business experts. There are three principal improvements:

- The three tasks obtained from external calls are removed, since the business experts consider that these tasks do not represent any of the organization's business activities, i.e. these are related to the technical dimension of the system.

  ```
  Remove ['getCustomer];
  Remove ['getUnits'];
  Remove ['insert'];
  ```

- A gateway is added to merge the two branches opened after the first gateway.

  ```
  Add [ GATEWAY['sendProducts', 'gateway(CheckInventory)', END] ];
  ```

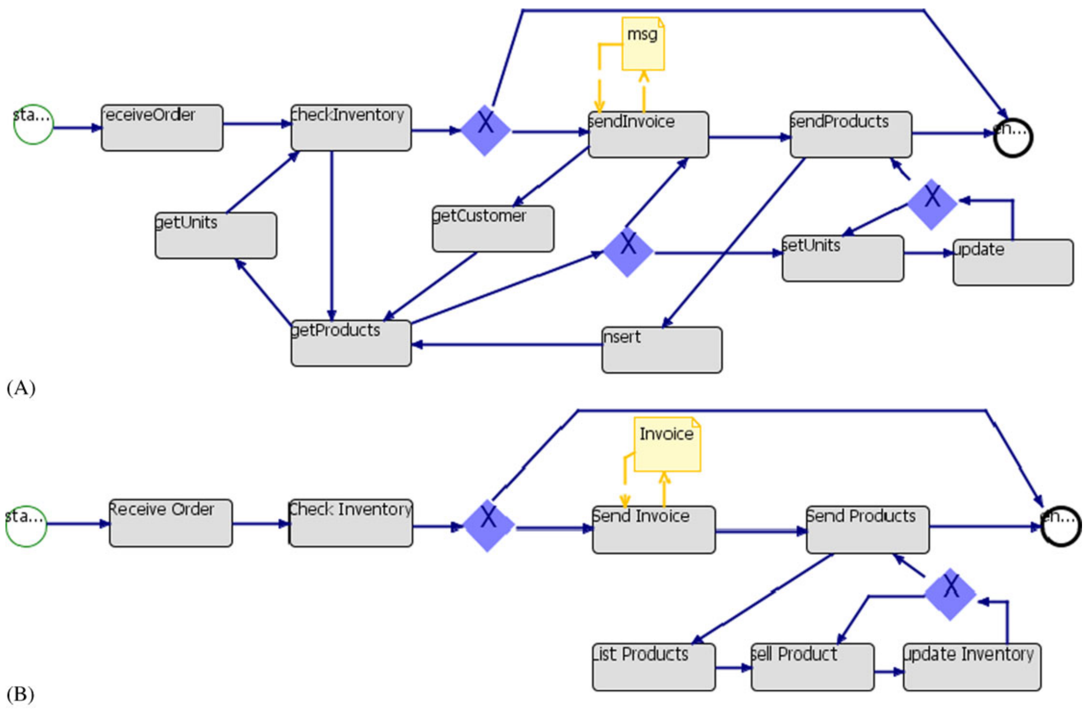


Figure 10. An example of business process models obtained for a product shipping system.

- The four remaining tasks are renamed by the business experts in order to fit the names to the organization's business activities.

```
Rename ['receiveOrder', 'Receive Order'];
Rename ['checkInventory', 'Check Inventory'];
Rename ['sendProducts', 'Send Products'];
Rename ['sendInvoice', 'Send Invoice'];
Rename ['getProducts', 'List Products'];
Rename ['update', 'Update Inventory'];
Rename ['setUnits', 'Sell Product'];
Rename ['msg', 'Invoice'];
```

Finally, Figure 10(B) shows the business process model that represents a part of the organization's behavior through the business process recovery from the piece of source code presented at the beginning of this example (see Figure 4, left-hand side).

5. A SUPPORTING TOOL

An *ad hoc* tool has been developed to semiautomate the proposed procedure. This tool facilitates the adoption of the recovery procedure as well as the execution of the case study. The tool is based on the Eclipse platform and uses four key technologies. The first technology is *JavaCC*, which is a parser generator for Java language [45]. The second technology is EMF (Eclipse Modeling Framework) [46]. This Eclipse framework makes it possible to build specific metamodels according to the *ECORE* meta-metamodel, the metamodel proposed by the Eclipse platform to define metamodels. EMF also provides tools to produce source code from *ECORE*-based metamodels to automatically build model viewers and editors. GMF (Graphical Modeling Framework) is additionally used together with EMF to generate graphical model editors. The third technology is XMI (XML Metadata Interchange), which defines the manipulation and interchange of models through XML

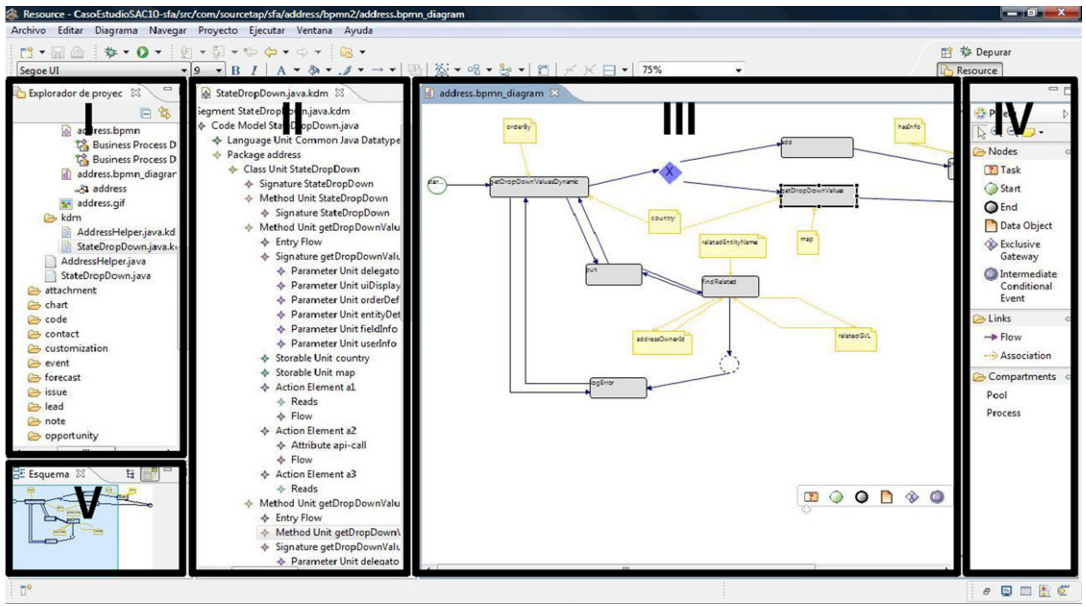


Figure 11. The tool to support the business process archeology method.

[47]. All the models involved in the procedure are thus made persistent by means of an XMI file. Finally, the fourth technology is *MediniQVT* [48], an open source framework which is a model transformation engine for QVT-Relations.

The tool architecture is divided into three different modules aligned with each of the transformations defined in the procedure. The first module supports the L0-to-L1 transformation. It thus consists of a static analyzer of Java source code, which is built though *JavaCC* from the EBNF (*Extended Backus–Naur Form*) grammar of Java 1.5. This module takes a Java file as input and then generates an XMI file as the output that represents the Java code model, a PSM model in L1 (see Figure 11(I)).

The second module executes a QVT transformation to generate the KDM model in L2 from the previous Java code models using *MediniQVT*. This module also has a tree view editor for KDM models and was built through EMF (see Figure 11(II)).

The third module also executes the QVT transformation of the L2-to-L3 transformation. Moreover, a graphical editor for BPD at L3 was developed using EMF/GMF (see Figure 11(III)), which also provide an outline view to make the work easier (see Figure 11(V)). This editor therefore allows business experts to modify and refine the business process models according to the proposed set of manual operations in order to fit the first version of business processes to the real-life organization's behavior. For this purpose, the tool provides a palette of business elements (see Figure 11(IV)).

6. CASE STUDY

This section presents a case study to validate the feasibility of the proposed business process recovery procedure by applying it to an e-government system. In order to automatically apply the proposal to the system, a tool that supports the entire depicted business process recovery procedure has been developed. The case study is carried out by following the protocol for planning, conducting and reporting case studies proposed by Brereton *et al.* [18]. The following sections present the details of the main stages defined in the formal protocol: background, design, case selection, case study procedure, data collection, analysis and interpretation, and validity evaluation.

Table III. Research questions of the case study.

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

6.1. Background

The previous research on this subject must first be identified. The related work presented in Section 2 shows other proposals for recovering business processes from legacy information systems, and compares these proposals with our procedure, considering four criteria. Our particular procedure is (i) a model-driven framework (more specifically, an ADM-based framework) for recovering business processes, which (ii) uses static analysis as the recovery technique, and (iii) source code as the main software artifact in addition to business expert information. Moreover, (iv) the case study is carried out by following a formal protocol for case studies [18].

The *object of study* is the proposed business process recovery procedure, and the *purpose of this study* is the evaluation of specific properties of the proposed procedure related to effectiveness and efficiency.

Bearing in mind the object and purpose of the study, the main research question (hereafter, *MQ*) addressed by the study is the following: *Can the recovery procedure properly obtain business processes from legacy systems*? In this question, the adverb *properly* means to obtain business processes in an effective and efficient way. In order to find out if *MQ* is true, Table III shows the additional research questions that are identified from the *MQ* question. The additional research question *AQ1* is established in order to evaluate whether the recovered business processes faithfully represent the organization's business behavior. That is, *AQ1* aims to evaluate the effectiveness of the procedure. Moreover, the additional research question *AQ2* (see Table III) evaluates whether the proposed business process recovery procedure obtains business processes efficiently.

6.2. Design

This case study consists of a single case, i.e. it focuses on a single legacy information system. The case study also considers several analysis units within the case. This study can therefore be considered as an *embedded* case study according to the classification proposed by Yin [49]. The analysis units are the different source code packages of the legacy information system, which is the independent variable of the study. Each source code package is represented in a model in L1 and L2 and is then transformed into a business process model in L3 through the third transformation. Despite the fact that there is a business process for each code package, the business process models obtained can be joined or split by means of the business experts' manual intervention according to the L2-to-L3 transformation.

The study consists of the analysis of each business processes recovered in order to answer the questions established in Table III. Some measures are therefore established to provide quantitative answers to the proposed research questions. First, in order to evaluate the effectiveness of the proposed procedure through the *AQ1* question, the study proposes the use of two measures: *Precision* and *Recall*. These measures were designed for information retrieval scenarios, although they are usually applied to other mining scenarios like business process recovery among others [50].

On the one hand, the *Precision* measure indicates the amount of relevant recovered elements within the set of recovered elements in a business process model. An element is considered relevant if this element faithfully represents the organization's business operations or business behavior in the real world. On the other hand, the *Recall* measure represents the amount of relevant recovered elements of the total of relevant elements (recovered and not recovered), which depicts the organization's entire business operation. While source code package is the independent variable, these two measures are dependent variables in the study.

The study considers the *task* element as the score unit in order to apply these measures in a business process recovery scenario. The *Precision* measure (1) is therefore defined as the number of true relevant tasks divided by the total number of relevant (recovered) tasks, i.e. the sum of true relevant tasks and false relevant tasks that were incorrectly recovered. Moreover, the *Recall* measure (2) is defined as the number of true relevant tasks divided by the total number of relevant tasks, i.e. the relevant tasks of the business process and other tasks that should have been recovered but were not recovered. Although *Precision* and *Recall* measures are adequate, there is an inverse relationship between them. The extraction of conclusions with an isolated evaluation of these metrics is consequently very difficult. These measures are therefore usually combined into a single measure known as an *F-measure* (3), which consists of a weighted harmonic mean of both measures.

$$PRECISION = \frac{|\{relevant\ tasks\} \cap \{recovered\ tasks\}|}{|\{recovered\ tasks\}|} \tag{1}$$

$$RECALL = \frac{|\{relevant\ tasks\} \cap \{recovered\ tasks\}|}{|\{relevant\ tasks\}|} \tag{2}$$

$$Fmeasure = \frac{2 \cdot PRECISION \cdot RECALL}{PRECISION + RECALL} \tag{3}$$

We also use the business expert opinion to discover which recovered tasks are relevant or not in order to evaluate the proposed measures. In this respect, the definition of *relevant task* is a key factor in facilitating the business experts' work, and it must be defined *a priori*. Despite the fact that the evaluation of measures focuses only on the business tasks, the *relevant task* definition implicitly considers other business elements, since the execution flow must not be ignored. The *relevant task* definition establishes a set of four conditions. Condition $C1$ specifies that the task must represent a real-life business operation within the organization. For instance, the task named '*parseDate*' obtained from an auxiliary Java method does not represent any valuable business activity. This condition must not be evaluated by considering task names, since these names are inherited from legacy code and they may be biased as regards the real business activity names provided by business experts. For example, the task named *receiveOrder*, despite its name is not exactly *Receive Product Orders*, it represents the valuable business activity in charge of receiving product orders. Condition C2 ensures that all the recovered tasks preceding the evaluated task must be relevant tasks. In order to meet C2, there can be no non-relevant tasks with a sequence flow to the evaluated task. In a similar way, Condition C3 ensures that all the following tasks must be directly (or indirectly) recovered relevant tasks. The conditions C2 and C3 check the control flow of business process models focusing on recovered tasks one by one. Finally, Condition C4 specifies that all the *Data Object* elements read and written by the evaluated task must also be recovered.

Second, in order to answer question *AQ2*, this study evaluates the total time spent on executing each transformation. The transformation time values are automatically measured for each analysis unit of the study by the tool developed. The time values obtained are analyzed with regard to the total number of elements built into each specific business process model. The purpose is to discover if the proposed recovery procedure can be scalable to a larger legacy system. Both the size of the business process models and the time values are also considered as the dependent variables of this study.

6.3. Case selection

The case selection is a key stage in the case study planning, which aims to select a good and suitable case to be studied. Table IV presents the five criteria established to select the most appropriate software system. *Cr1* guarantees that the legacy system selected is an information system that supports an organization or company's business operation. This criterion discards, for example, embedded systems or real-time systems. *Cr2* ensures that the legacy system will be a real-life

Table IV. Criteria for case selection.

Id	Criteria for case selection
Cr1	It must be an enterprise system
Cr2	It must be a real-life system
Cr3	It must be a legacy system
Cr4	It must be of a size not less than 100 KLOC
Cr5	It must be a Java-based system

system that is deployed in a production environment. *Cr3* ensures that the selected system really is a legacy information system. To evaluate this criterion the time in production is not used, since it is not a good measure. Instead of production time, *Cr3* considers the amount of modifications in the system that alter the business processes, i.e. the system modifications related to the *adaptive* and *perfective* maintenance according to [51]. *Cr4* ensures that the system is not a *toy program*, since it defines a threshold of 100 000 lines of source code. Finally, *Cr5* guarantees that the system is based on the Java platform, since both the proposed recovery procedure and the tool were developed for Java-based systems.

After evaluating several available systems according to the previous criteria, the legacy information system that was selected to be studied was *eAdmin*, an e-government system. This system supports the electronic administration of a Spanish regional ministry of Housing and Urban Development. This system automates all the services offered by the government and their document management, thus meeting the *Cr1* criterion. The first release of the system was moved to the production stage 26 months ago, thus meeting the *Cr2* criterion. During this time, the same development team of the *eAdmin* system has made three medium modifications (versions 1.1, 1.2, and 1.3), a large modification (version 2.0), and two medium modifications (versions 2.1 and 2.2). This ensures compliance with the *Cr3* criterion. From a technological point of view, *eAdmin* is a Web application and its architecture is separated into three layers: presentation, business, and persistence. The technology used to develop the presentation layer was *JSP* (*Java Server Pages*), *Java* for business layer, and *Oracle* together with *JDBD-ODBC* for the persistence layer. The total size of the legacy system is 320.2 KLOC. Criteria *Cr4* and *Cr5* are therefore also met.

6.4. Case study procedure

In addition to the design and case selection of the case study, the execution procedure of the study must also be planned. The execution is aided by the tool developed to support the procedure. The case study procedure defines the followings steps:

1. After some meetings between the staff of the candidate organizations and researchers, a legacy information system is selected according to the case selection criteria. The business expert needed to perform the manual post verification is also selected in this step. In this study, this person is the *chief information officer* of the Spanish regional ministry of Housing and Urban Development, since this person has the highest level of expertise with regard to the organization's business activities.
2. The legacy system under study is implanted in an experimental environment: the source code is deployed in a Web server, the database schema is built through the database scripts, the initial data is loaded, etc.
3. The different business process models are obtained from the legacy source code using the tool that supports the procedure (cf. Section 5). The execution hardware environment consists of a computer with a RAM memory of 4 GB and two processors each of 2.67 GHz.
4. The first sketch of business processes obtained through the model transformation is improved by business experts. They fit the preliminary business processes with the reality of the organization, i.e. they can add tasks that should have been recovered but were not recovered, or remove tasks erroneously recovered. After business expert intervention, the accuracy of business process models is evaluated by comparing each preliminary business process and its

related business process enhanced by business experts. We obtain the value of the proposed measures like precision and recall by scoring the differences between the preliminary and enhanced business processes.

5. The key information related to the generation of business processes (cf. step 3), along with the business expert intervention (cf. step 4), is collected for each analysis unit according to the data collection plan (cf. Section 6.5).
6. The data collected in the previous step are analyzed and interpreted to draw conclusions in order to answer the research questions. Section 6.6 shows the outgoing results obtained in this case study. Finally, the case study is reported and the feedback is given to the organization and research community.

6.5. *Data collection*

The data to be collected and the data sources must be defined before starting the execution of the case study. The following information is thus recorded for each analysis unit (see Table V): (i) the number of source code files in each package; (ii) the number of business process models obtained after the L2-to-L3 transformation; (iii) the total number of elements in the business process model (the size of the model); (iv) the total number of recovered tasks in order to evaluate after the Precision and Recall metrics; (v) the total transformation time in milliseconds. Finally, the bottom part of Table V summarizes all the information by means of the total, mean and standard deviation for each data column.

The right-hand side of Table V also summarizes the manual interventions by the business expert, and thus presents (i) the ID number of each preliminary business process model and (ii) the inter-model operation(s) carried out in each model or between models (e.g. rename, remove, join and split models).

Table VI shows (i) the number of tasks recovered (before manual intervention); (ii) the number of recovered relevant tasks, i.e. the number of tasks that the business expert marked as correct; (iii) the number of recovered non-relevant tasks, i.e. the tasks removed from the business process since they did not represent a business activity; (iv) *Precision* and (v) *Recall* values for each final business process; and (vi) the harmonic mean between them. The measures are calculated with data from the previous columns of this table. Table VI also presents (i) the total number of business elements in each final business process and (ii) the total transformation times, which are derived through aggregation from the data collected in Table V.

6.6. *Analysis and interpretation*

After the data have been collected, they are analyzed in order to draw conclusions. The analysis should obtain the evidence chains from data to answer the research questions. In order to answer the question *AQ1*, Figure 12 shows the box chart for *Precision* and *Recall* measures. The chart shows the density distributions for the set of final business processes. The mean of the distribution of the *Precision* measure (0.66) is lower than the mean of *Recall* measure (0.91). Moreover, the standard deviation of the *Recall* measure is close to 0 (0.05), i.e. the central values of the *Recall* distribution are more concentrated around the mean than the central values of the *Precision* measure (0.13).

The interpretation of these values is the following. On the one hand, a higher *Recall* value means that the proposed procedure recovers a higher number of relevant business elements, i.e. it recovers most of the tasks from the current business processes. On the other hand, the *Recall* value contrasts with the low *Precision* value, which means that the number of non-relevant tasks is very high with regard to the recovered tasks. A low *Precision* and a high *Recall* mean that the preliminary business processes were obtained as large business processes with almost all the relevant tasks (high *Recall* value), but they were obtained with a great amount of non-relevant tasks (low *Precision* value) that were removed by business experts. The low *Precision* value, i.e. the significant amount of non-relevant tasks, is due to the fact that some tasks are obtained almost directly from the source code and they are related to the technical nature, and do not therefore represent any piece of the embedded business knowledge.

Table V. Data collected in the case study.

Package	# source code files	# BPMN models	# Elements	# Tasks	Transf. time (ms)	Preliminary BP Id	Manual intervention in business process models
d.seventhLaw	32	1	347	73	2001	1	J[22, 23], RN [‘Administration’]
d.admin	75	1	485	77	5032	2	R
d.spfPurchase	44	1	556	70	5134	3	J[24], RN[‘Social Protection Floor MGMT’]
d.enclose	11	1	344	27	2548	4	J[25], RN[‘Document MGMT’]
d.rent	140	1	1855	265	22 850	5	J[26],RN[‘Rental House MGMT’]
d.rnDocument	104	1	922	189	10 195	6	J[28], RN[‘Renovation Document Registration’]
d.common	82	1	604	121	5416	7	R
d.applicant	53	1	1340	191	12 804	8	J[29], RN[‘House Applicant MGMT’]
d.ppFiles	57	1	622	117	4690	9	J[30], RN[‘Personal File MGMT’]
d.ruralHouse	38	1	595	107	8427	10	J[31], RN[‘Rural House MGMT’]
d.developer	64	1	795	135	6348	11	J[32],RN[‘Developer MGMT’]
d.renovation	20	1	511	54	10 037	12	R
d.rnFacade	28	1	783	113	8666	13	J[34], RN[‘Renovation MGMT’]
d.grants	39	1	471	85	4325	14	J[35], RN[‘Emancipation Grant MGMT’]
d.secondHand	38	1	476	74	3369	15	J[37], RN[‘Second-Hand House MGMT’]
d.user	10	1	216	48	1040	16	R
pdfhandler	6	1	490	55	2364	17	R
eRegistration	8	1	82	14	4385	18	R
security	11	1	145	30	1077	19	R
util	25	1	340	92	1795	20	R
web	9	1	322	67	1518	21	R
w.seventhLaw	10	1	76	12	503	22	J[1,23], RN[‘Administration’]
w.admin	88	1	313	49	2267	23	J[1,22], RN[‘Administration’]
w.spfPurchase	9	1	143	27	761	24	J[3], RN[‘Social Protection Floor MGMT’]
w.enclose	6	1	15	6	176	25	J[4], RN[‘Document MGMT’]
w.rent	29	1	126	25	533	26	J[5],RN[‘Rental House MGMT’]
w.help	1	1	69	1	86	27	R
w.rnDocument	27	1	125	15	543	28	J[6],RN[‘Renovation Document Registration’]
w.applicant	15	1	154	22	406	29	J[8],RN[‘House Applicant MGMT’]
w.ppFiles	20	1	112	17	559	30	J[9],RN[‘Personal File MGMT’]
w.ruralHouse	7	1	79	11	1956	31	J[10],RN[‘Rural House MGMT’]
w.developers	22	1	29	11	686	32	J[11],RN[‘Developer MGMT’]
w.renovation	3	1	16	4	186	33	R

Table V. *Continued.*

Package	# source code files	# BPMN models	# Elements	# Tasks	Transf. time (ms)	Preliminary BP Id	Manual intervention in business process models
w.rnFacade	12	1	143	19	1420	34	J[13],RN['Renovation MGMT']
w.grants	12	1	196	37	976	35	J[14],RN['Emancipation Grant MGMT']
w.session	2	1	1	0	110	36	R
w.secondHand	11	1	181	31	933	37	J[15],RN['Second-Hand House MGMT']
w.user	13	1	88	19	560	38	R
w.validator	2	1	12	2	109	39	R
Total	1183	39	14179	2312	136791		
Mean	30.3	1.0	363.6	59.3	3507.5		
Std. deviation	31.6	0.0	384.3	59.9	4556.9		

The results obtained are usual since there is an inverse relationship between *Precision* and *Recall* measures. Figure 13 represents the inverse relationship between both measures, and shows the results obtained (high recall, but low precision). The *Precision* value should, ideally, always be 1.0 for any *Recall* value but, according to [52], this is not possible in practice (see Figure 13). Thus, owing to the fact that the relationship between both measures has an inverse nature, it is possible to increase one at the cost of reducing the other. Indeed, the proposed procedure could reduce its *Recall* value by recovering fewer tasks, at the cost of reducing the number of non-relevant recovered tasks, i.e. increasing the *Precision* value. This hypothetical result is more desirable than the obtained result, since the *Precision* and *Recall* values would be more balanced.

Indeed, the *F-measure* (3) can be considered as a special case of the general F_α*-measure* (4), where α is 1. The α value is used to weight the *Precision* in relation to *Recall*, and the selected F_1*-measure* (3) thus signifies that both *precision* and *recall* are equally important. A good balance between *Precision* and *Recall* would therefore be better to obtain a higher F_1*-measure* value

$$F_\alpha = \frac{(1+\alpha)\cdot PRECISION \cdot RECALL}{\alpha \cdot PRECISION + RECALL} \tag{4}$$

In order to jointly evaluate both measures, Figure 14 shows the F_1*-measure* values obtained, along with the *Precision* and *Recall* values for each final business process. The F_1*-measure* has a mean of 0.76 and a standard deviation of 0.10.

These values are additionally compared with reference values from other experiences with model recovery in the literature, such as those of [53–55]. We found reports of *Precision* and *Recall* values close to 60%, and these were our benchmark values. The values obtained for our measures ($Precision = 66\%$, $Recall = 91\%$ and $F_1 - measure = 76\%$) were therefore clearly above 60%, the reference value. Question *AQ1* can consequently be positively answered, i.e. the proposed procedure makes it possible to recover business processes from legacy information systems with adequate accuracy levels. However, the *F-measure* value might be slightly improved by obtaining more balanced *Precision* and *Recall* values.

The transformation time was also analyzed in order to answer question *AQ2*. After applying the proposed recovery procedure, 39 preliminary business process models were obtained with an average size of 363.6 elements (or 59.3 tasks) per BPD (see Table VI). The total time was 137 s (2'28" approximately), and the processing rate is thus equal to 2.65 elements per second. Finally, the time spent on processing each task was 2.31 s.

Despite the fact that the mean of the *Precision* is a medium value in this study, the time wasted as a result of recovering non-relevant tasks is low. The total time if we consider only the final business process models was, therefore, 103 s, and the time spent on recovering non-relevant processes thus represents only 24.8%.

Table VI. Data of final business process models.

Business process model	# Recovered Tasks (RcT)	# Recovered relevant tasks (RcRvT)	# Recovered non-relevant task (RcNRvT)	# Non-recovered relevant tasks (NRcRvT)	Precision (RcRvT/RcRvT+ +RcNRvT)	Recall (RcRvT/RcRvT+ NRcRvT)	F-Measure (harmonic) (mean)	# Elements	Transf. time (ms)
Administration	134	69	65	10	0.51	0.87	0.65	736	4771
Social Protection Floor MGMT	97	64	33	6	0.66	0.91	0.77	699	5895
Document MGMT	33	23	10	3	0.70	0.88	0.78	359	2724
Rental House MGMT	290	221	69	9	0.76	0.96	0.85	1981	23383
Renovation Document Reg.	204	169	35	3	0.83	0.98	0.90	1047	10738
House Applicant MGMT	213	190	23	9	0.89	0.95	0.92	1494	13210
Personal File MGMT	134	74	60	9	0.55	0.89	0.68	734	5249
Rural House MGMT	118	63	55	12	0.53	0.84	0.65	674	10383
Developer MGMT	146	87	59	15	0.60	0.85	0.70	824	7034
Renovation MGMT	132	91	41	3	0.69	0.97	0.81	926	10086
Emancipation Grant MGMT	122	62	60	7	0.51	0.90	0.65	667	5301
Second-Hand House MGMT	105	76	29	9	0.72	0.89	0.80	657	4302
Mean	144	99.08	44.92	7.92	0.66	0.91	0.76	899.8	8589.7
Std. deviation	65.5	60.30	18.94	3.73	0.13	0.05	0.10	437.1	5635.0

Copyright © 2011 John Wiley & Sons, Ltd.

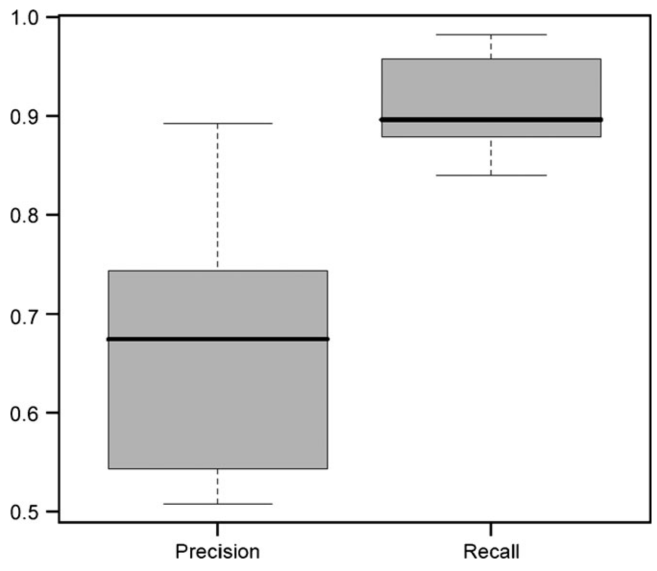


Figure 12. The box chart for *Precision* and *Recall* measures.

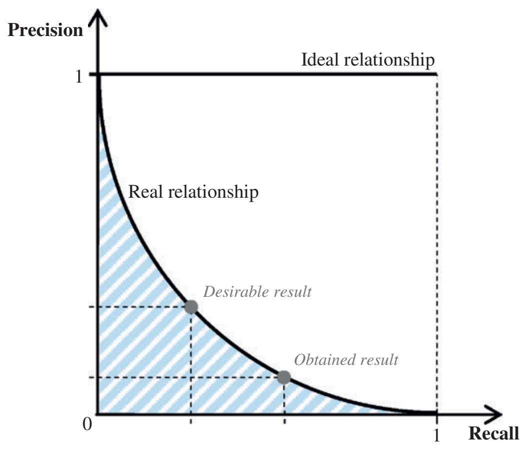


Figure 13. Relationship between *Precision* and *Recall* measures, and obtained results.

A total time of 137 s for the proposed transformations seems feasible for the selected case, since the size of the *eAdmin* is 320.2 *KLOC*. Nevertheless, the scalability of the procedure must be evaluated. Under the hypothesis that the time complexity of the procedure theoretically is linear, we established a linear regression model to check it and find out if the proposal is scalable. The linear regression model considers the transformation time as a dependent variable and the size of the analysis units as an independent variable. Figure 15 shows the scatter chart of size/time, which represents two charts considering (a) the total number of elements (circle points) and (b) the number of tasks (square points) as the size of each business process model. This chart also shows the regression lines, which in both cases present a positive linear relationship between the model size and the time spent on model transformations.

Moreover, Figure 15 shows the correlation coefficient R^2, which is the degree to which the real values of the dependent variable are close to the predicted values, i.e. how much the points are fitted to the regression line. In this study, the correlation coefficient was 0.89, considering the total elements as the model size, and 0.79 considering the task number as size. Since the correlation coefficient value is between 0 and 1 for a positive linear relationship, these values are very high. This signifies that the proposed linear regression model is suitable to explain the data obtained in

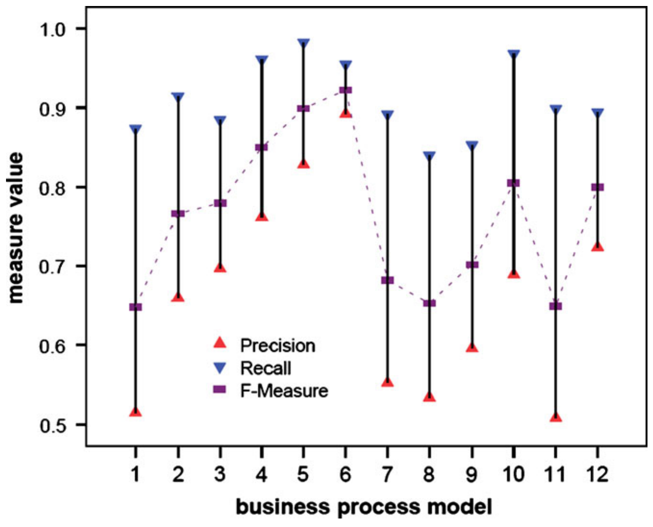


Figure 14. F-Measure values for the final business processes.

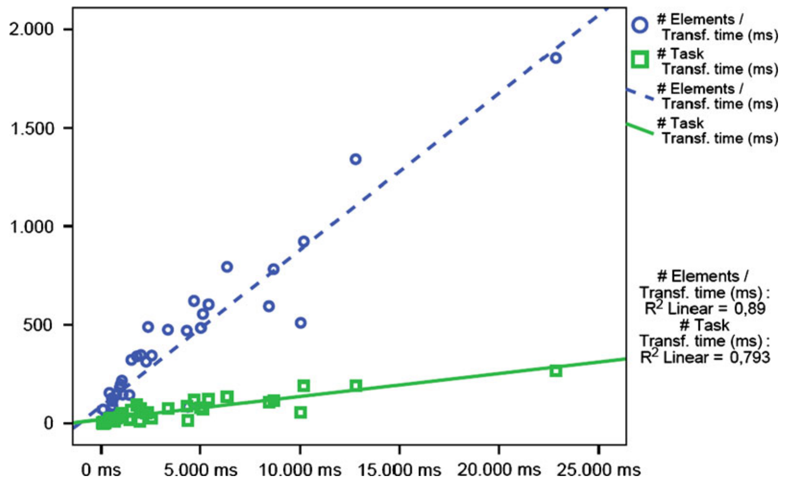


Figure 15. The size/time scatter chart.

this study, i.e. there is no quadratic or exponential relationship between transformation time and model size. The increase in time for larger systems will consequently be linear, and the time will thus be assumable.

This procedure is semiautomatic, thus the manual effort carried out by business experts must also be analyzed besides the machine computation time. The selected business expert took 96.3 min to refine the first version of the business processes obtained by the tool. That effort focused on the refinement of 39 preliminary business process models into 12 final models and the refinement implied a total reduction concerning tasks over 37%. As a consequence, the total time spent on recovering the embedded business processes through the proposed procedure was 98.8 min. The manual time spent on this study is greater than the computational time. However, the manual time related to our proposal would be less than the time spent on a manual business process redesign from scratch. In addition, the manual time will probably be linear with respect to the size of legacy information systems in the same way as the computational time.

In conclusion, the *AQ2* question can therefore be answered as true, and the main research question *MQ* is also answered as true, i.e. the proposed business process recovery procedure makes it possible to obtain business process models from legacy information systems in an effective and efficient way.

6.7. Validity evaluation

Finally, the validity of the results obtained in the case study must be evaluated. This stage evaluates whether the results are true and not biased for the whole population to which we would generalize the results. This section thus shows the threats to the validity of this case study. According to [56] there are three main types of validities: internal, construct, and external validity.

6.7.1. Internal validity. There is no large population that makes it possible to obtain statistically representative results, although a clear trend for the proposed measures was identifiable in this case study. However, there are two determining factors related to obtaining the results presented here. First, the tool used to obtain the business processes could be a factor that affects the transformation time. This means that the measure values might be different if the business processes were obtained with another tool supporting the same recovery procedure. Second, if this study is replicated with other cases involving a different business expert, the results concerning *Precision* and *Recall* measures might have some deviations as a result of the business expert's manual intervention, since this person provides his/her subjective point of view.

6.7.2. Construct validity. The measures of the case study were adequate to measure the variables and answer the research questions appropriately. The *Precision* and *Recall* measures were reused from the information retrieval field, in which these metrics have an adequate maturity level. These measures also allow us check whether or not the business processes obtained faithfully represent the organization's business operative, since these values can be compared with reference values. Moreover, the time and size measures allow us to answer the research question regarding the scalability of the proposal.

6.7.3. External validity. Finally, *external validity* is concerned with the generalization of the results. This study considers traditional legacy information systems as the whole population. In this respect, the results obtained could be generalized to this population. However, the specific platform of the selected case is a threat that should be noted. The results are thus strictly extended to those legacy information systems based on object-oriented languages like *Java*.

7. CONCLUSIONS AND FUTURE WORK

This paper has addressed the problem of business process recovery from legacy systems in order to preserve valuable business knowledge. We have therefore presented a recovery procedure as a possible solution, which is framed in MARBLE (an ADM-based framework). The proposed procedure is characterized by three main features: (i) it focuses on legacy source code as the main source of knowledge, (ii) it uses static analysis as a reverse engineering technique to extract the information needed; and (iii) it follows the model-driven development principles, i.e. it considers different models at different abstraction levels and a set of model transformations between them. Furthermore, we have also developed a Java supporting tool in order to automate the procedure and provide mechanisms for its adoption.

In order to validate the proposal, we have presented a real-life case study as a key contribution of this paper. The selected case is an e-government system based on Java, which is used in the application of the proposed recovery procedure. The case study aims to evaluate both the effectiveness and efficiency of the proposal. In order to evaluate the effectiveness, we use the *Precision* and *Recall* measures. We believe that these metrics are adequate to estimate the degree to which the recovered business processes faithfully represent the organization's business operation. Moreover, in order to evaluate the efficiency, the case study evaluates the recovery time with regard to the size of each business model. This evaluation allows us to check the scalability of our proposal to large legacy systems.

The results obtained in the study show that the proposed procedure is suitable for recovering business processes from legacy information systems from the point of view of effectiveness and efficiency. That is, the procedure is able to obtain business processes that depict an organization's business behavior. Moreover, the case study was planned, conducted, and reported by following a formal protocol for conducting case studies. This study can consequently be adequately replicated in the future using other legacy information systems.

Our future work will address the identified threats to the validity. On the one hand, we hope to use meta-analysis to contrast the results of this case with the results obtained for the same study concerning other legacy information systems. On the other hand, we hope to replicate this study with legacy systems based on other platforms or languages in order to compare and generalize (if possible) the results obtained.

In addition to improving the validation of the proposed procedure, we also hope to refine it. In this respect, we aim to incorporate the dynamic analysis in order to extract more and valuable business knowledge. Other software artifacts could also be considered as databases or user interfaces in the future.

ACKNOWLEDGEMENTS

This work was supported by the FPU Spanish Program; by the R+D projects funded by JCCM: ALTAMIRA (PII2I09-0106-2463), INGENIO (PAC08-0154-9262) and PRALIN (PAC08-0121-1374), and the PEGASO/MAGO project (TIN2009-13718-C02-01) funded by MICINN and FEDER.

REFERENCES

1. Weske M. *Business Process Management*: *Concepts*, *Languages*, *Architectures*. Springer: Berlin/Heidelberg, Leipzig, Alemania, 2007; 368.
2. Jeston J, Nelis J, Davenport T. *Business Process Management*: *Practical Guidelines to Successful Implementations*, (2nd edn). Butterworth-Heinemann (Elsevier Ltd.): NV, U.S.A., 2008; 469.
3. Heuvel W-JVD. *Aligning Modern Business Processes and Legacy Systems*: *A Component-Based Perspective* (*Cooperative Information Systems*). The MIT Press: Cambridge, MA, 2006.
4. Visaggio G. Ageing of a data-intensive legacy system: Symptoms and remedies. *Journal of Software Maintenance* 2001; **13**(5):281–308.
5. Lehman MM. Program evolution. *Information Processing and Management* 1984; **20**(1–2):19–36.
6. Koskinen J, Ahonen J, Lintinen H, Sivula H, Tilus T. Estimation of the business value of software modernizations. Information Technology Research Institute, University of Jyväskylä, 2004.
7. Paradauskas B, Laurikaitis A. Business knowledge extraction from legacy information systems. *Information Technology and Control* 2006; **35**(3):214–221.
8. Canfora G, Di Penta M. New frontiers of reverse engineering. *Future of Software Engineering*. IEEE Computer Society: Silver Spring, MD, 2007.
9. Chikofsky EJ, Cross JH. Reverse engineering and design recovery: A taxonomy. *IEEE Software* 1990; **7**(1):13–17.
10. Khusidman V, Ulrich W. Architecture-driven modernization: Transforming the enterprise. DRAFT V.5, OMG, 2007; 7. Available at: http://www.omg.org/docs/admtf/07-12-01.pdf [12 October 2007].
11. Sneed HM. Estimating the costs of a reengineering project. *Proceedings of the 12th Working Conference on Reverse Engineering*. IEEE Computer Society: Silver Spring, MD, 2005; 111–119.
12. OMG. Architecture-driven modernization standards roadmap, 2009. Available at: http://www.omg.org/docs/admtf/07-12-01.pdf [29 October 2009].
13. Newcomb P. Architecture-driven modernization (ADM). *Proceedings of the 12th Working Conference on Reverse Engineering*. IEEE Computer Society: Silver Spring, MD, 2005.
14. Pérez-Castillo R, García-Rodríguez de Guzmán I, Ávila-García O, Piattini M. MARBLE: A modernization approach for recovering business processes from legacy systems. *International Workshop on Reverse Engineering Models from Software Artifacts* (*REM'09*). Simula Research Laboratory Reports: Lille, France, 2009; 17–20.
15. Lewis GA, Smith DB, Kontogiannis K. A research agenda for service-oriented architecture (SOA): Maintenance and evolution of service-oriented systems. Software Engineering Institute, 2010; 40.
16. ISO/IEC, ISO/IEC DIS 19506. Knowledge discovery meta-model (KDM), v1.1 (Architecture-Driven Modernization), 2009; 302. Available at: http://www.iso.org/iso/catalogue_detail.htm?csnumber=32625 [19 March 2009].

17. Pérez-Castillo R, García-Rodríguez de Guzmán I, Piattini M. Implementing business process recovery patterns through QVT transformations. *International Conference on Model Transformation* (*ICMT'10*). Springer: Málaga, Spain, 2010; 168–183.
18. Brereton P, Kitchenham B, Budgen D, Li Z. Using a protocol template for case study planning. *Evaluation and Assessment in Software Engineering* (*EASE'08*), Bari, Italia, 2008; 1–8.
19. Daga A, Cesare SD, Lycett M, Partridge C. An ontological approach for recovering legacy business content. *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences* (*HICSS'05*), *Track 8*, *vol. 8*. IEEE Computer Society: Silver Spring, MD, 2005; 224.1.
20. OMG, Why do we need standards for the modernization of existing systems? (OMG ADM Task Force), 2003.
21. Kazman R, Woods SG, Carrière SJ. Requirements for integrating software architecture and reengineering models: CORUM II. *Proceedings of the Working Conference on Reverse Engineering* (*WCRE'98*). IEEE Computer Society: Silver Spring, MD, 1998.
22. Miller J, Mukerji J. MDA Guide Version 1.0.1. OMG, 2003; 62. Available at: www.omg.org/docs/omg/03-06-01.pdf [21 April 2010].
23. OMG. QVT. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, 2008. Available at: http://www.omg.org/spec/QVT/1.0/PDF [21 March 2009].
24. OMG. Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM), v1.1, 2009; 308. Available at: http://www.omg.org/spec/KDM/1.1/PDF/ [2 January 2009].
25. Moyer B. Software Archeology. Modernizing Old Systems. Embedded Technology Journal, 2009. Available at: http://adm.omg.org/docs/Software_Archeology_4-Mar-2009.pdf [1 June 2009].
26. OMG. Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM), v1.0, 2008; 310. Available at: http://www.omg.org/docs/formal/08-01-01.pdf [7 August 2008].
27. Cai Z, Yang X, Wang W. Business process recovery for system maintenance—An empirical approach. *25th International Conference on Software Maintenance* (*ICSM'09*). IEEE CS: Edmonton, Canada, 2009; 399–402.
28. Eisenbarth T, Koschke R, Simon D. Locating features in source code. *IEEE Transactions on Software Engineering* 2003; **29**(3):210–224.
29. Wang X, Sun J, Yang X, He Z, Maddineni S. Business rules extraction from large legacy systems. *Proceedings of the Eighth Euromicro Working Conference on Software Maintenance and Reengineering* (*CSMR'04*). IEEE Computer Society: Silver Spring, MD, 2004.
30. Zou Y, Lau TC, Kontogiannis K, Tong T, McKegney R. Model-driven business process recovery. *Proceedings of the 11th Working Conference on Reverse Engineering* (*WCRE 2004*). IEEE Computer Society: Silver Spring, MD, 2004; 224–233.
31. Ghose A, Koliadis G, Chueng A. Process discovery from model and text artefacts. *IEEE Congress on Services* (*Services'07*), Salt Lake City, UT, 2007; 167–174.
32. Paradauskas B, Laurikaitis A. Business knowledge extraction from legacy information systems. *Journal of Information Technology and Control* 2006; **35**(3):214–221.
33. Pérez-Castillo R, García-Rodríguez de Guzmán I, Caballero I, Polo M, Piattini M. PRECISO: A reengineering process and a tool for database modernisation through web services. *24th Annual ACM Symposium on Applied Computing* (*SAC'09*), Hawaii, U.S.A., 2009; 2126–2133.
34. Di Francescomarino C, Marchetto A, Tonella P. Reverse engineering of business processes exposed as web applications. *13th European Conference on Software Maintenance and Reengineering* (*CSMR'09*). IEEE Computer Society: Fraunhofer IESE, Kaiserslautern, Germany, 2009; 139–148.
35. Günther CW, van der Aalst WMP. A generic import framework for process event logs. *Business Process Intelligence Workshop* (*BPI '06*) (Lecture Notes in Computer Science, vol. 4103). Springer: Berlin, 2007; 81–92.
36. Ingvaldsen JE, Gulla JA. Preprocessing support for large scale process mining of SAP transactions. *Business Process Intelligence Workshop* (*BPI'07*) (Lecture Notes in Computer Science, vol. 4928). Springer: Berlin; 2008; 30–41.
37. van der Aalst W, Reijers H, Weijters A. Business process mining: An industrial application. *Information Systems* 2007; **32**(5):713–732.
38. Cornelissen B, Zaidman A, Deursen AV, Moonen L, Koschke R. A systematic survey of program comprehension through dynamic analysis. *IEEE Transactions on Software Engineering* 2009; **35**(5):684–702.
39. OMG. Business Process Model and Notation (BPMN) 2.0. Object Management Group, 2009; 496.
40. Müller HA, Jahnke JH, Smith DB, Storey M-A, Tilley SR, Wong K. Reverse engineering: A roadmap. *Proceedings of the Conference on the Future of Software Engineering*. ACM: Limerick, Ireland, 2000.
41. Pérez-Castillo R, García-Rodríguez de Guzmán I, Ávila-García O, Piattini M. Business process patterns for software archeology. *25th Annual ACM Symposium on Applied Computing* (*SAC'10*). ACM: Sierre, Switzerland, 2010; 165–166.
42. Aalst WMPVD, Hofstede AHMT, Kiepuszewski B, Barros AP. Workflow patterns. *Distributed and Parallel Databases* 2003; **14**(3):5–51.
43. Zdun U, Hentrich C, Dustdar S. Modeling process-driven and service-oriented architectures using patterns and pattern primitives. *ACM Transactions on the Web* 2007; **1**(3):14.
44. Zhao L, Macaulay L, Adams J, Verschueren P. A pattern language for designing e-business architecture. *Journal of Systems and Software* 2008; **81**(8):1272–1287.
45. Open Source Initiative, JavaCC 4.2. A parser/scanner generator for java, 2009. Available at: https://javacc.dev.java.net/ [3 February 2010].

46. EMF, Eclipse Modeling Framework Project. The Eclipse Foundation. IBM Corporation, 2009. Available at: http://www.eclipse.org/modeling/emf/ [15 December 2009].
47. OMG. XML Metadata Interchange. MOF 2.0/XMI Mapping, v2.1.1. OMG, 2007. Available at: http://www.omg.org/spec/XMI/2.1.1/PDF [7 May 2007].
48. ikv++, Medini QVT. ikv++ technologies ag, 2008. Available at: http://www.ikv.de/index.php?option=com_content&task=view&id=75&Itemid=77 [25 October 2010].
49. Yin RK. Case study research. *Design and Methods* (3rd edn). Sage: London, 2003.
50. Pradel M, Bichsel P, Gross TR. A framework for the evaluation of specification miners based on finite state machines. *26th IEEE International Conference on Software Maintenance* (*ICSM'10*), Timioara, Romania, 2010.
51. ISO/IEC, ISO/IEC 14764:2006. Software Engineering—Software Life Cycle Processes—Maintenance, 2006. Available at: http://www.iso.org/iso/catalogue_detail.htm?csnumber=39064 [23 August 2006].
52. Davis J, Goadrich M. The relationship between Precision-Recall and ROC curves. *Proceedings of the 23rd International Conference on Machine Learning*. ACM: Pittsburgh, PA, 2006; 233–240.
53. Lucrédio D, Fortes RPM, Whittle J. MOOGLE: A model search engine. *11th International Conference on Model Driven Engineering Languages and Systems*. Springer: Toulouse, France, 2008; 296–310.
54. Ye Y, Fischer G. Supporting reuse by delivering task-relevant and personalized information. *24th International Conference on Software Engineering*. ACM: Orlando, FL, 2002; 513–523.
55. Garcia VC, Lucrédio D, Durão FA, Santos ECR, Almeida ESD, Fortes RPDM, Meira SRDL. From specification to experimentation: A software component search engine architecture. *Ninth International Symposium on Component-Based Software Engineering* (*CBSE 2006*). Springer: Västerås, Sweden, 2006; 82–97.
56. Wohlin C, Runeson P, Höst M, Magnus OC, Regnell B, Wesslén A. *Experimentation in Software Engineering*: *An Introduction*. Kluwer Academic Publishers: Dordrecht, 2000; 204.

Copyright © 2011 John Wiley & Sons, Ltd.