2015 12th International Conference on Information Technology - New Generations

# An Approach to Business Process Recovery from Source Code

Luiz A. Pacini Rabelo, Antonio F. do Prado and Wanderley L. de Souza
*Department of Computer Science*
*Federal University of Sao Carlos (UFSCar)*
*Sao Carlos, SP, Brazil*
*Email: {luiz.pacini, prado, desouza}@dc.ufscar.br*

Luís F. Pires
*Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)*
*University of Twente*
*P.O. Box 217, 7500 AE Enschede, The Netherlands*
*Email: l.ferreirapires@utwente.nl*

***Abstract*—Over time Business Process has become an asset for organization since it allows managing what happens within their environments. It is possible to automate some activities of the business process using information systems and accordingly decrease the execution time and increase the production. However, information systems often suffer maintenance over time and become obsolete and a re-engineering process is necessary. In this case, the business knowledge, located more accurately the reality in source code, should be maintained. Thereof, this paper propose an approach to support the business process recovery from source code. For this purpose, the approach uses KDM standard with a set of heuristic rules to identify relevant code elements to the business layer. As result, the models are generated according to the BPMN specification that, together with other artifacts, provide more subsidies to the professionals involved. To evaluate the effectiveness of the approach, a case study was performed in an Academic Management System.**

***Keywords*-Business Process; Knowledge Discovery; Software Re-engineering; Reverse Engineering;**

## I. Introduction

Business Process represent a set of activities performed in an organization with the purpose of reaching a Business Objective, i.e. a product or a service. These business process provide a way to map the business goals about the best way to be performed in different application domains and, therefore, has become an asset in organization [1].

Moreover, business process support the development process of information systems, constituting the stage of requirements analysis and defining essential requirements to achieve business goals [2].

Currently, most organizations use information systems to automate some activities of the business process to provide more agility in performing them [3]. Nonetheless, information systems are not immune to the effects of time and along several maintenance, tend to become obsolete in their technologies [4]. It occurs due to the business domain and the software domain are subject to constant change and independent evolutions. While in the business domain, the business process are adapted to meet the requirements of customers, in the software domain, new features are added to information systems to enhance competitiveness [5].

Along the changes, the information systems become increasingly lagged and without documentation to assist the maintenance [5]. For that reason, it is often necessary to perform a software re-engineering process in order to update it with new technologies and maintain the coherence between the business process and the source code [6].

Nevertheless, perform a re-engineering process to rebuild and information system is not trivial. For this, it is necessary recover the business knowledge (i.e., the business process) embedded into the system and this knowledge is found more accurately the reality in source code, compared with the system documentation and other artifacts [7].

The recovered business process may be used by organizations in two ways: (I) to provide for business specialists a better understanding of the operation of the organization; and (II) to develop a new improved system to reduce the effects of time. The improved system supports the current business process and improves the Return On Investment (ROI) once its useful life is increased [3].

Although great progress has been made in the Software Re-engineering area, especially in the phase of Reverse Engineering, many challenges to improve this scenario remain. The reverse engineering process can obtain abstract representation of the system, including the business knowledge representation, however, re-engineering projects rarely reach the abstraction level of the business, reaching usually only the system design level [8].

Therefore, tools for source code analysis become alternatives to support the business process recovery embedded in the source code and, thus, the Software Re-engineering.

Once this need is supplied, obtain reductions of time and effort expended to perform this task, allowing redirect resources to other tasks of the re-engineering process. Furthermore, the generated artifacts (business process models) together with other artifacts (source code, documentation, etc.) provide more subsidies to the involved professionals for understanding what the system does and thus boost the re-engineering process.

Thereby, this paper proposes an approach to support the business process recovery embedded in the information system source code. The approach uses the Knowledge Discovery Meta-model standard (KDM) together with a set of heuristic rules to identify relevant code elements to the

978-1-4799-8828-0/15 $31.00 © 2015 IEEE
DOI 10.1109/ITNG.2015.64

business layer and the Business Process Model and Notation standard (BPMN) to represent them.

The remainder of the paper is organized as follows. Section II presents the main concepts and technologies used in the proposed approach; Section III describes the proposed approach to support the business process recovery from source code; Section IV evaluates the approach through a case study; Section V discusses the related work; and finally, Section VI concludes.

## II. Background

The following subsections introduces the two main concepts to have a better understanding about the proposal: the KDM and BPMN standards.

### A. Knowledge Discovery Metamodel

The Knowledge Discovery Meta-model (KDM) [11] is a standard defined by the Object Management Group (OMG) and has become an international standard ISO/IEC 19506 ADM KDM [12]. The KDM specification defines a meta-model for the modeling of different software artifacts involved in a legacy system information. The KDM meta-model provides an overview of the behavior, structure and data from legacy systems [12] and it is divided into layers that represent the logical and physical software assets at various levels of abstraction [11].

KDM meta-model addressed the main challenges of modernization of legacy information systems and it is the main standard among a set of standards defined by the OMG. The meta-models are represented in XML (eXtensible Markup Language) using the OMG standards, such as MOF (Meta-Object Facility) and XMI (XML Meta-data Interchange). Thus, the KDM allows: (I) store the facts about information systems in a compliant structure; (II) analyze and reason about KDM facts; (III) exchange KDM facts as XML documents; and (IV) build tools so that parsing source code in a given programming language to generate language-independent and vendor-neutral facts about information systems [9].

The MoDisco is an Open Source Eclipse Project that provides a Framework to the development of tools based on models and offers an implementation of the KDM specification [10].

### B. Business Process Model and Notation

The Business Process Model and Notation (BPMN) is a standard for modeling business process created by the Business Process Management Initiative (BPMI), whose main goal is to provide an easily understandable notation for all business users. A second equally important goal is to ensure that languages designed to the execution of business process, such as Business Process Execution Language (BPEL), can be visually expressed in a common notation [11].

The BPMN modeling is done using diagrams with a small set of graphical elements. The diagrams are simple and intuitive, however, able to represent complex processes. The notation organizes graphical aspects in specific categories, namely: Flow Objects, Connecting Objects, Swimlanes and Artifacts. Within the categories, variations and additional information can be added to support requirements for complexity without dramatically change the appearance of the diagram [12].

The jBPM is a suite of Business Process Management open source, developed in Java and distributed under Apache license, which allows modeling, execution and monitoring business process. Furthermore, jBPM allows also create business process models in three ways: (I) from a graphical editor; (II) from a XML file following the BPMN specification; and (III) from Java factories using the jBPM API [13].

## III. Proposed Approach

Based on the presented concepts and technologies, this paper proposes an approach to support the business process recovery from source code. The approach aims at providing a practical way to support the business process recovery embedded in the source code of information systems, providing a reduction of time and effort expended to accomplish this task. Additionally, the approach allows to represent the recovered business process models in BPMN, an intuitive graphical notation and easily understandable.

Figure 1 shows an overview of the proposed approach described in BPMN. The process consists of three activities: Model Discovery, Business Elements Identification and Business Process Representation.

The approach uses as input the source code of information systems. From the source code, the activity Model Discovery

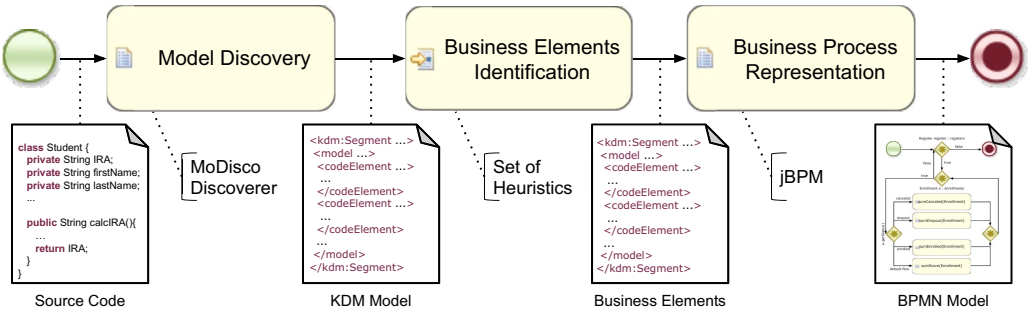

Figure 1. Overview of the Proposed Approach.

generates the KDM Platform-Independent Model (PIM). This model aims at portraying the source code in a structure based on XMI. From the PIM model, the activity Business Process Identification applies a set of heuristic rules to select the relevant code elements to business process. A list of business elements is obtained by this activity. This list is the own input PIM model, but modified to contain only business elements. The next activity, Business Process Representation, uses this list of business elements to build the BPMN models though the BPM API. Finally, we have as result the BPMN models that describe the business process embedded in the information system source code.

### A. Model Discovery

The Model Discovery activity aims to derive one or more models of a given information system and represent them in a uniform and homogeneous way through models. Each models conforms to a given meta-model and has a correspondence one-to-one with the source code. The MoDisco is used as a support in this activity to generate a PIM model from a given programming language source code. The generated model contain the complete Abstract Syntax Tree (AST) of application source code, the resolution of the links between the identifiers in the source code and the relationship between model elements [14].

```
1 package br.ufscar.prograd.business; // 1
2
3 public class Register { // 2
4   public Double sumScore(Enrollment e) {...}    // 3
5   public Integer sumEnrolled(Enrollment e) {...}
6   public Integer sumCanceled(Enrollment e) {...}
7   public Integer sumDropout(Enrollment e) {...}
8   public void calculateAPI(List<Register> registers) { // 4
9     ...
10    for (Register register : registers) {      // 5
11      ...
12      List<Enrollment> enrollments = enrollment.getEnrollments(register);
13      for (Enrollment e : enrollments) {       // 6
14        if (e.getStatus() == "canceled")       // 7
15          sumCanceled += this.sumCanceled(e);
16        else if (e.getStatus() == "dropout")
17          sumDropout += this.sumDropout(e);
18        else if (e.getStatus() == "enrolled")
19          sumEnrolled += this.sumEnrolled(e);
20        else
21          sumScore += this.sumScore(e);
22      }
23      Double API = (sumScore / sumEnrolled)     // 8
24      * (2 - ((2 * sumDropout + sumCanceled) / sumEnrolled));
25    }
26  }
27 }
```

Figure 2. A snippet of an Academic Management System Source Code.

Figure 2 illustrates an example of a snippet of an Academic Management System Source Code. This example illustrates the `Register` class (2), responsible for the enrollments register of the university. The class belongs to `br.ufscar.prograd.business` package (1) and contains some implemented methods (3). Among the implemented methods, we have `calculateAPI` method (4) that calculates the Academic Performance Index (API) of students with active enrollments in classroom courses. The `calculateAPI` method has two loops, one to register (5) and one for enrollments (6). For each enrollment of each register, is performed the sum of measures (7) that will be used later to calculate the `API` (8).

```
1 <kdm:Segment>
2  <model xsi:type="code:CodeModel" name="ProGradWeb">
3   <codeElement xsi:type="code:Package" name="business"> <!-- 1 -->
4    <codeElement xsi:type="code:ClassUnit" name="Register"> <!-- 2 -->
5     <codeElement xsi:type="code:MethodUnit" name="sumScore"></> <!-- 3 -->
6     <codeElement xsi:type="code:MethodUnit" name="sumEnrolled"></>
7     <codeElement xsi:type="code:MethodUnit" name="sumCanceled"></>
8     <codeElement xsi:type="code:MethodUnit" name="sumDropout"></>
9     <codeElement xsi:type="code:MethodUnit" name="calculateAPI"> <!-- 4 -->
10     <codeElement xsi:type="action:BlockUnit">
11      <codeElement xsi:type="action:ActionElement" name="foreach"> <!-- 5 -->
12       <codeElement xsi:type="action:BlockUnit">
13        <codeElement xsi:type="action:ActionElement" name="foreach"> <!-- 6 -->
14         <codeElement xsi:type="action:BlockUnit">
15          <codeElement xsi:type="action:ActionElement" name="if"> <!-- 7 -->
16           <codeElement xsi:type="action:ActionElement" name="EQUALS"></>
17           <codeElement xsi:type="action:ActionElement" name="method
18             invocation"></>
19           <codeElement xsi:type="code:Value" name="string" ext="canceled"></>
20          </codeElement>
21          ...
22        </codeElement>
23        <codeElement xsi:type="action:ActionElement" name="variable
24            declaration">
25         <codeElement xsi:type="code:StorableUnit" name="API">...</> <!-- 8 -->
26        </codeElement>
27     ...
28   </codeElement>
29  </model>
30 </kdm:Segment>
```

Figure 3. A snippet of KDM Model generated through a snippet of an Academic Management System Source Code.

Figure 3 illustrates a snippet of KDM model generated from the source code of the previous example. For each element of source code, it creates a corresponding element in KDM model. For example, the Register class (2) in Figure 2, is represented by a `CodeElement` with type `code:ClassUnit` and also marked as (2); a Method such as calculateAPI method (4) in Figure 2, is represented by a `CodeElement` with type `code:MethodUnit` and also marked as (4); a Loop For, such as the (5), is also represented by a `CodeElement` but with type `action:ActionElement` and kind `foreach` and marked as (5); and so on.

### B. Business Elements Identification

The Business Elements Identification activity applies a set of heuristic rules into KDM model to obtain the relevant business elements for the business process. These heuristic rules are according with Business Process Patterns [15], which defines standards of business process implementations, i.e., specifies the structure of source code elements (described by KDM elements) that generate specific business structures in BPMN models. These heuristic rules are divided into two categories, inclusion rules and exclusion rules.

The inclusion rules are responsible for select all the code elements relevant to the business layer. While the exclusion rules are responsible for exclude the code elements irrelevant to the business layer. Thus, a code element relevant to business layer should satisfy at least one inclusion rule and simultaneously satisfy no exclusion rule. Accordingly, is possible select more efficiently the business elements.

Some inclusion heuristic rules that compose the set are:

**BPD Skeleton:** creates the root structure of the model. For each KDM code model, a new diagram is created, and for each package of KDM code model, a pool element with a nested process in the diagram is built.

**Sequence:** takes any callable piece of code from the KDM code model and maps them into tasks in the diagram. In ad-

dition, the sequence of calls to callable units is transformed into a set of sequence flows in the same order between the tasks built from the callable unit respectively.

**Branching:** transforms each conditional jump into an exclusive gateway. Typically those exclusive conditional branches are related with the if...then...else or switch clauses.

**Collaboration:** transforms each call to external callable unit (i.e. API libraries or external components) into an auxiliary task.

**Input/Output Data:** build a data object for each input/output data involved in a callable unit in the KDM code model.

**Start:** build a start event linked with the first task of the diagram, i.e., the task that start the execution of program or application.

**Termination:** build an end event linked with all end tasks, i.e., the tasks that do not have any outgoing sequence flow.

**Exception:** transforms each call to callable unit under any exception into an error intermediate event.

Additionally to these inclusion heuristic rules, the set also consists of exclusion rules, such as Graphical Import rule. This rule exclude any class that import graphical libraries (`javax.swing.*`, `java.awt.*`, etc.), once it is responsible for graphical objects and is irrelevant to the business layer.

Another exclusion rule is Database Access, that exclude any class that import database libraries (`com.mysql.jdbc.*`, `net.sf.hibernate.*`, etc.). Moreover, this rule build a pool to represent the system's database and, for each method invocation that belongs to these class, a communication flow is created between the caller task and the database pool.

At the end of this activity, a list of business elements is obtained. This list is the own input PIM model, but modified to contain only relevant code elements to business layer. These business elements can be also represented by the model illustrated in Figure 3 and will be used for the next activity to build the BPMN models.

### C. Business Process Representation

The Business Process Representation activity aims to generate understandable artifacts that describe business process. This activity uses the list of business elements generated by the previous activity and map these elements to BPMN elements. To support this activity is used the jBPM API, an API that allows building BPMN models by Java programming factories.

Figure 4 illustrates an example a Java factory of a snippet of an Academic Management System, which corresponds to the mapping of business elements resulting from previous activity. This example shows how to map business elements to Java factories using jBPM API. A start event is represented by a `startNode`, as well as an end event is an `endNode`. There are two types of gateways: `splitNode` and `joinNode`. Activities are represented as `actionNode` and sub-processes as `subProcessNode`. For more information, see [13].

```
1 RuleFlowProcessFactory factory = RuleFlowProcessFactory
2         .createProcess("calculateAPI");
3 factory
4  // Header
5  .name("Register::calculateAPI") // 2, 4
6  .version("1.0")
7  .packageName("br.ufscar.prograd.business") // 1
8  // Nodes
9  .startNode(1).done()
10 .splitNode(2).type(TYPE_XOR) // 5
11   .constraint(3, "true", "code", "Java", "Register register : registers")
12   .constraint(10, "false", "code", "Java", "Register register : registers").done()
13 .splitNode(3).type(TYPE_XOR) // 6
14   .constraint(4, "true", "code", "Java", "Enrollment e : enrollments")
15   .constraint(2, "false", "code", "Java", "Enrollment e : enrollments").done()
16 .splitNode(4).type(TYPE_XOR) // 7
17   .constraint(5, "true", "code", "Java", "e.getStatus() == \"canceled\"")
18   .constraint(6, "true", "code", "Java", "e.getStatus() == \"dropout\"")
19   .constraint(7, "true", "code", "Java", "e.getStatus() == \"enrolled\"")
20   .constraint(8, "true", "code", "Java", "default flow").done()
21 .actionNode(5).name("sumCanceled(Enrollment)").done()
22 .actionNode(6).name("sumDropout(Enrollment)").done()
23 .actionNode(7).name("sumEnrolled(Enrollment)").done()
24 .actionNode(8).name("sumScore(Enrollment)").done()
25 .joinNode(9).type(TYPE_XOR).done()
26 .actionNode(10).name("API = ...").done() // 8
27 .endNode(11).done()
28 // Connections
29 .connection(1,2).connection(2,3).connection(2,11)...;
30
31 RuleFlowProcess process = factory.validate().getProcess();
```

Figure 4. A Java factory of a snippet of an Academic Management System Source Code.

Similarly to the previous examples, the `splitNode` (5) in this figure correspond to the same code segment marked as (5) in Figures 2 and 3. As well as the actionNode (8) in this figure correspond to the same code segment marked as (8) in Figures 2 and 3, and so on.

The Figure 5 illustrates graphically the business process described in Figure 4.

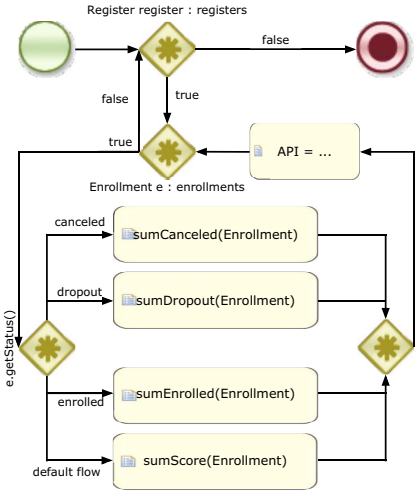


Figure 5. A BPMN model of a snippet of an Academic Management System Source Code.

## IV. Case Study

This section presents a case study to evaluate the effectiveness of the proposed approach. The case study was performed in academic domain of Academic Management

System (ProGradWeb[1]) into the Federal University of So Carlos (UFSCar). The system was developed in Java and consists of 60.9 KLOC, organized into 7 packages and 167 classes.

The effectiveness was measured by precision and recall. Precision (1) measures the accuracy and fidelity of the recovered business processes, defined as the number of recovered relevant tasks divided by the total number of recovered tasks. Recall (2) measures the completeness of the recovered business processes, defined as the number of recovered relevant tasks divided by the total number of relevant tasks.

$$\text{Precision} = \frac{|\{\text{relevant tasks}\} \bigcap \{\text{recovered tasks}\}|}{|\{\text{recovered tasks}\}|} \quad (1)$$

$$\text{Recall} = \frac{|\{\text{relevant tasks}\} \bigcap \{\text{recovered tasks}\}|}{|\{\text{relevant tasks}\}|} \quad (2)$$

Although the Precision and Recall measures are sufficient, there is an inverse relationship between them that makes it difficult to evaluate. Thus, these metrics are usually combined into a single measure, known as F-measure (3), which consists of a weighted harmonic average of the two measurements.

$$\text{F-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

Table I summarizes the obtained results from execution of the proposed approach on the effectiveness, i.e., precision, recall and F-measure. These measures are between [0,1] and can vary from one system to another, although the tendency is that recall be higher than precision.

Table I
OBTAINED RESULTS IN THE CASE STUDY.

| | Business Process Model |
|---|---|
| # Recovered Tasks | 257 |
| # Recovered Relevant Tasks | 204 |
| # Recovered Non-relevant Tasks | 53 |
| # Non-recovered Relevant Tasks | 24 |
| Precision | 0.79 |
| Recall | 0.89 |
| F-measure | 0.84 |

According to Bender Method [16], precision and recall above 0.72 and 0.88 respectively, are considered very high. In addition, F-measure is considered very high if equal or greater than 0.75. Therefore, as shown in Table I, the obtained results are considered very high, which proves the effectiveness of the proposed approach.

## V. RELATED WORK

The problem of the preservation of business knowledge is not new. Several years back reverse engineering applied to the business process recovery of information systems has been researched. Some works address the business process recovery from the static analysis of several software artifacts, while other works address the business process recovery from the dynamic analysis. There are also studies that address both analyzes (static and dynamic), and others that dealing the problem more specifically according to their own needs.

*Wang et al.* [17] provides a framework for the extraction of business rules from large information systems based on the static program slicing. However, the framework is not able to represent the business knowledge in a comprehensible form. *Zou et al.* [5] developed a model-driven framework based on a set of heuristic rules that parse the source code and extract business processes. However, the proposal is not independent of platform or language. *Paradauskas e Laurikaitis* [7] recover the business knowledge through inspection of data stored in databases, together with the application source code. The evaluation of the proposal, however, was performed through the development of a controlled example. *Ghose et al.* [18] use text-based queries executed in documentations to extract the business knowledge, but this approach was validated using a simple example.

*Eisenbarth et al.* [19] present a technique for locating business features based on dynamic analysis. However, this proposal does not extract the business knowledge according with business processes notation, and demonstrates applicability by an example. *Cai et al.* [20] proposes an approach that combines the requirements acquisition with the technique of static and dynamic analysis to extract complex business processes that are triggered by external agents. *Prez-Castillo et al.* [21] provide the MARBLE, a tool based on the ADM standard which performs transformations between models to recover business processes. The tool enables both static analysis and dynamic analysis of source code. *Di Francescomarino et al.* [22] recover business processes by dynamic analysis of the Web GUI applications that run while user navigation. Moreover, this study provides a clustering algorithm to minimize the obtained business processes.

The main handicap these works is that most efforts are ad-hoc proposals which are developed without standardization and for specific platforms, technologies and contexts. This lack of formalization and standardization creates problems to replicate these techniques in large-scale projects. This problem could be solved adopting, for example, standards such as ADM / KDM, which provide a means to homogenize models independently of platform.

## VI. CONCLUSION

This paper presented an approach to support the business processes recovery from source code in order to preserve the valuable business knowledge embedded in information systems. The proposed approach uses the static analysis, a technique of reverse engineering to extract information from information systems source code. Furthermore, the approach is based on analysis of the KDM model, which allows the

[1]http://www.prograduweb.ufscar.br/

description of the source code of information systems in a platform independent model.

The Approach consists of three activities: Model Discovery, Business Elements Identification e Business Process Representation; and allowed abstracting business processes embedded in information systems source code and represent them in BPMN models. The recovered BPMN models together with other artifacts (source code, documentation, etc.), provide larger subsidies to involved professionals understand more clearly what the system does and, consequently, supports the software modernization process. Thus, the approach contributes to extend the lifetime of information systems and thereby helps to improve ROI.

The feasibility of the approach was confirmed by performing a case study applied to a real system in the academic domain, showing success to recover business process from source code. Although, the proposed approach did not recover all relevant tasks, the case study demonstrated the effectiveness by means of the precision and recall metrics, with values $0.79$ and $0.89$ respectively. This review allowed verifying the scalability of the proposed approach to large information systems, i.e., information systems with more than 100 KLOC.

As future work, will be developed a Domain Specific Language (DSL) to describe the heuristics. Thus, the approach becomes refinable and easily maintained, since it would allow the inclusion and/or exclusion of new rules. Furthermore, the case study will be replicated for systems developed to other platforms or languages in order to compare the results.

## ACKNOWLEDGMENTS

Our thanks to CAPES and FAI-UFSCar for financial sponsorship.

## REFERENCES

[1] M. Weske, *Business Process Management: Concepts, Languages, Architectures.* Springer Berlin Heidelberg, 2007, p. 368.

[2] J. Jeston and J. Nelis, *Business Process Management: Practical Guidelines to Successful Implementations*, 2nd ed. Butterworth-Heinemann (Elsevier Ltd.): NV, U.S.A., 2008, p. 469.

[3] R. Pérez-Castillo, I. G.-R. d. Guzmán, M. Piattini, and Á. S. Places, "A Case Study on Business Process Recovery Using an e-Government System," *Softw. Pract. Exper.*, vol. 42, no. 2, pp. 159–189, Feb. 2012.

[4] G. Visaggio, "Ageing of a Data-intensive Legacy System: Symptoms and Remedies," *Journal of Software Maintenance*, vol. 13, no. 5, pp. 281–308, Oct. 2001.

[5] Y. Zou, T. C. Lau, K. Kontogiannis, T. Tong, and R. McKegney, "Model-Driven Business Process Recovery," in *Proceedings of the 11th Working Conference on Reverse Engineering*, ser. WCRE '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 224–233.

[6] R. Pressman, *Software Engineering: A Practitioner's Approach*, ser. McGraw-Hill series in computer science. McGraw-Hill Higher Education, 2010.

[7] B. Paradauskas and A. Laurikaitis, "Business knowledge extraction from legacy information systems," *Information Technology and Control*, vol. 35, no. 3, pp. 214–221, 2006.

[8] V. Khusidman and W. Ulrich, "Architecture-driven modernization: Transforming the enterprise," *DRAFT V.5, OMG*, 2007.

[9] R. Pérez-Castillo, I. G.-R. De Guzman, and M. Piattini, "Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems," *Computer Standards & Interfaces*, vol. 33, no. 6, pp. 519–532, 2011.

[10] H. Bruneliere, J. Cabot, F. Jouault, and F. Madiot, "MoDisco: a generic and extensible framework for model driven reverse engineering," in *Proceedings of the IEEE/ACM international conference on Automated software engineering*. ACM, 2010, pp. 173–174.

[11] M. Owen and J. Raj, *BPMN and Business Process Management: Introduction to the New Business Process Modeling Standard*, 2003.

[12] S. A. White, "Introduction to BPMN," *IBM Cooperation*, vol. 2, no. 0, p. 0, 2004.

[13] JBoss, "jBPM Documentation v6.1," JBoss, Tech. Rep., 2014.

[14] V. Cosentino, "A model-based approach for extracting business rules out of legacy information systems," Ph.D. dissertation, Ecole des Mines de Nantes, 2013.

[15] R. Pérez-Castillo, I. G. R. de Guzmán, and M. Piattini, "On the use of patterns to recover business processes," in *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, 2010, pp. 165–166.

[16] R. Bender, "Quantitative risk assessment in epidemiological studies investigating threshold effects," *Biometrical Journal*, vol. 41, no. 3, pp. 305–319, 1999.

[17] X. Wang, J. Sun, X. Yang, Z. He, and S. Maddineni, "Business rules extraction from large legacy systems," in *Software Maintenance and Reengineering, 2004. CSMR 2004. Proceedings. Eighth European Conference on*. IEEE, 2004, pp. 249–258.

[18] A. K. Ghose, G. Koliadis, and A. Cheung, "Process discovery from model and text artefacts," *Faculty of Informatics-Papers*, p. 571, 2007.

[19] T. Eisenbarth, R. Koschke, and D. Simon, "Locating features in source code," *Software Engineering, IEEE Transactions on*, vol. 29, no. 3, pp. 210–224, 2003.

[20] Z. Cai, X. Yang, and X. Wang, "Business process recovery for system maintenance—An empirical approach," in *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*. IEEE, 2009, pp. 399–402.

[21] R. Pérez-Castillo, "MARBLE: Modernization approach for recovering business processes from legacy information systems," in *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, Sept 2012, pp. 671–676.

[22] C. Di Francescomarino, A. Marchetto, and P. Tonella, "Reverse engineering of business processes exposed as web applications," in *Software Maintenance and Reengineering, 2009. CSMR'09. 13th European Conference on*. IEEE, 2009, pp. 139–148.