SURVEY AND PROPOSAL OF A METHOD FOR BUSINESS RULES IDENTIFICATION IN LEGACY SYSTEMS SOURCE CODE AND EXECUTION LOGS

Wanderley Augusto Radaelli Junior, Gleison Samuel do Nascimento and Cirano Iochpe
Informatics Institute, Federal University of Rio Grande do Sul, 9500 Bento Gonçalves Av., Porto Alegre, Brazil

Keywords: Legacy systems modernization, Reverse engineering, Business process, Business rules, Source code manipulation, Execution logs mining.

Abstract: Computer systems implement business processes from different organizations. Among the currently operating computer systems, much of it is classified as legacy system. Typically, legacy systems are complex applications that are still active, due to the high cost of modernization and a high degree of criticality. In recent years, were published several works addressing the importance of legacy systems modernization, emphasizing the extraction of the business process model implemented in these systems. Within this context, a key step is to extract knowledge from source code and / or systems execution logs, aiming to use this information in reverse engineering processes. In this work are presented and analyzed methods based on source code manipulation and system's execution logs mining, which can be used to extract knowledge from legacy systems, prioritizing business rules identification. A comparison between the two different approaches is presented, as well as their positive and negative characteristics. Our results include a list of desired features and a proposal of a method for legacy systems reverse engineering and business rules identification.

1 INTRODUCTION

Legacy systems are information systems which accomplish useful tasks within an organization, but were developed based on technologies which are no longer in use. These systems include information and procedures which are fundamental for the operation of the organization. However, to keep a legacy system running is a complex, error-prone and costly task (Pressman, 2001). In particular its program code can be obsolete, hard to understand, and poorly documented.

To reduce these problems, many organizations invest in rewriting their legacy systems based on modern technologies (Pressman, 2001). In (Nascimento, Iochpe, Thom, Reichert, 2009), we propose a sequence of rewriting steps that should be followed in order to restructure legacy code on both *Business Process Management (BPM)* and *Service Oriented Architecture (SOA)*. In short, this method encompasses two subsequent phases. The first one consists of identifying and translating the dynamic behavior of a legacy system into business processes models. Source code is analyzed and its behavior is translated to directed graphs representing business processes models. In a second phase those business processes models are implemented and can be executed by using a BPMS (*Business Process Management System*) (Weske, 2007).

To identify the business processes which are implicitly implemented within a legacy system constitutes a major challenge of any rewriting method that relies on BPM and SOA technologies. In order to discover business processes within legacy code, in (Nascimento, Iochpe, Thom, Reichert, 2009) we present a technique that consists of analyzing the legacy source code in order to identify its dynamic behavior and to use this knowledge later on to construct directed graphs that represent the business processes that exist within it.

Business rule is a key concept to the proposed legacy code analysis technique. A business rule is an abstracting statement that describes either a behavioral or an execution constraint of an information system (Ross, 1997). A business rule defines either an action or control point of the execution flow of an information system.

Radaelli Junior W., do Nascimento G. and Iochpe C..
SURVEY AND PROPOSAL OF A METHOD FOR BUSINESS RULES IDENTIFICATION IN LEGACY SYSTEMS SOURCE CODE AND EXECUTION LOGS.
DOI: 10.5220/0003483802070213
In *Proceedings of the 13th International Conference on Enterprise Information Systems* (ICEIS-2011), pages 207-213
ISBN: 978-989-8425-55-3

Usually, business rules are identified and applied to the logical specification of information systems. By the implementation of an information system, these statements are translated into source code fragments. Therefore, it is correct to say that the at least some parts of a legacy system code implement business rules that define as well as produce its dynamic behavior during system execution.

In a similar way, a business process model, too, can be specified through business rules (Knolmayer, Endl, Pfahrer, 2000). By the implementation of a business process model, the business rules that must be represented within it are mapped onto individual process components or process fragments. Thus, we propose to use business rules as invariants in the mapping of parts of legacy code onto process model fragments of equivalent behavior.

In this work, we present a set of business rule types to describe the behavior of different parts of a legacy code. This set was found sufficient in all case studies we have carried out until now in the context of the BPM/SOA based rewriting technique we are investigating.

For every business rule type contained in the specified subset and for every programming language, a syntactical structure can be defined that allows for the representation of rules of that type in any source code written in that specific programming language. Therefore, one can create a set of rule type templates in any programming language he/she selects. Each one of the rule templates can then be used either manually or automatically in order to identify rule instances within a source code written in the respective programming language.

An important step of the BPM/SOA based rewriting technique we are investigating is the identification of business rules that are represented in the legacy source code by applying a set of rule type templates as in a pattern matching procedure onto the legacy source code.

In this context, in the literature, there are several studies addressing methods that assist the task of business rules identification in legacy systems. Most of these methods are based on reverse engineering processes (Nascimento, Iochpe, Thom, Reichert, 2009) (Kalsing, Nascimento, Iochpe, Thom, Reichert, 2010a) (Erlikh, 2000).

There are two main approaches to reverse engineer systems, aiming at further business rules identification:

1. Methods that involve source code manipulation: usually, these methods are based on static analysis of the source code.
2. Methods that involve data mining: these methods are based on the application of data mining algorithms, which analyze execution logs of the target system's.

This paper present and analyze methods which allow source code manipulation and / or system execution logs mining, in order to identify our business rules set that make up the business process model of the system being analyzed. To reduce the need of human intervention and to identify business rules automatically or semi-automatically constitute an important contribution to the research area.
Thus, the main contributions of this paper are:

- Definition of the business rules set used in the rewriting method presented in (Nascimento, Iochpe, Thom, Reichert, 2009);
- An analysis about how the presented methods can be utilized to identify the business rules set used in the rewriting method presented in (Nascimento, Iochpe, Thom, Reichert, 2009);
- Presentation of a legacy systems reverse engineering hybrid method of business rules identification.

The remainder of this paper is organized as follows: section 2 shows methods of legacy systems reverse engineering used to identify business rules in source code and execution logs. Section 3 presents the business rules set that is used in the rewriting method presented in (Nascimento, Iochpe, Thom, Reichert, 2009). In Section 4 are listed some desired features for legacy systems reverse engineering methods. Section 5 contains a proposal of a legacy systems reverse engineering hybrid method suitable for the identification and extraction of the whole set of business rules listed in section 3. Finally, the conclusion outlines the main contributions of this research work.

2 RELATED WORK

In this section are presented and analyzed legacy systems reverse engineering methods, which can be used in knowledge extraction processes, emphasizing business rules identification.

2.1 Methods based on Source Code Manipulation

In order to identify business rules in legacy systems, methods based on source code manipulation, mainly, use a combination of techniques (e.g. program slicing), concepts (e.g. domain variables), and data

structures (e,g. abstract syntax tree - AST).

In (Chiang, 2006) the author presents a method of legacy systems reverse engineering, which uses program slicing technique as base for source code manipulation and business rules identification. For better application within large systems, the method presented, first classify the source code into three different categories, which are: user interface layer, data access layer and business rules layer. The author argues that this division simplifies the source code manipulation process. After the classification, dependency graphs are created to guide the program slicing process, which leads to business rules identification with human aid.

In (Wang, Zhou, Chen, 2008), is presented a method of legacy systems reverse engineering based on domain variables identification. The method also uses program slicing technique and dependency graphs. The article presents a classification for system variables, introducing the concept of pure domain variables, arguing that this kind of variable is generally directly related to business rules, referencing external data sources such as files or database connections. In the end of the process there is a validation step, in which the identified business rules candidates are presented for system experts.

Similarly, in (Wang, Sun, Yang, He, Maddineni, 2004), knowledge extraction and subsequent identification of business rules is based on domain variables management. The proposed method create dependency graphs and identify domain variables related to input/output operations. According to the authors, the method was proposed to overcome a limitation that exists in other methods, which make the identification and manipulation of domain variables manually. In large systems the proposed method should be implemented module by module, due to the large number of domain variables found.

A method for business rules identification and extraction is also presented in (Putrycz, Kark, 2007). The method consists of several steps, and the most important are: creation of an AST representing the system, extraction of information from the AST and creation of a knowledge base (code blocks, identifiers, conditional branches, loops, comments in source code). The final step correspond to extracted data validation, and is performed with the aid of system specialists, using a prototype developed by the authors, which evaluates the information contained in the knowledge base created earlier.

A semi automatic method of legacy systems reverse engineering, which uses a combination of AST, dependency graphs and program slicing technique is presented in (Paradauskas, Laurikaitidis, 2006). The method consists of eight steps, among which we can highlight the following: generation of AST, generation of dependency graphs, application of program slicing technique, final validation through human intervention in the end of process. The variant of program slicing applied depends on the type of the variable being analyzed: backward slicing to handle output variables, and forward slicing to handle input variables.

In (Huang, Tsai, Bhattacharya, Chen, Wang, Sun, 1996) is presented an interactive method for business rules identification within legacy systems source code. In the first step, the source code is parsed, resulting in an AST and a data dictionary. In the sequence, a dependency graph is created based on the AST. Using the DG and some heuristic rules presented in the paper, the variables present in the source code are classified. After this step, the most important variables are selected and guide the program slice technique application. The result is then presented in a user interface prototype, where the users can participate of the business rules identification and extraction process.

2.2 Methods based on Mining Execution Logs

In (Van Der Aalst, Reijers, Weijters, 2007) is proposed a method of legacy systems reverse engineering and modernization based on systems execution logs mining. The text is based on ideas present in ProM framework, designed by the authors. The main characteristics of the proposed method are: it is generic in the sense that it is not geared to the characteristics of the system being analyzed. Represents the information in ProM framework syntax, which is not trivial, specially for non-technical staff. More than one data mining algorithm is used. The main contributions of the paper are: authors present arguments and examples which justify the combination of various data mining algorithms, in order to obtain better results within mining large systems execution logs. Authors claim that interaction with system users is necessary to resolve doubts and validate the business rules extracted, because in some cases, the information in the log are not self-explanatory, making sense only to users with good experience in the system.

A method of reverse engineering system based on data mining is presented in (Stroulia, El-Ramly, Kong, Sorenson, Matichuk, 1999). The proposed method is based on mining logs of the user interaction with the system. The authors used the

naming traces of execution, rather than execution logs. The text is focused on the ideas presented by the project CelLEST, which is conducted by the authors. The main steps of the proposed method are: identify the business process model implemented by the system using Genetic Miner algorithm. The interaction patterns are derived from the sequence of screens accessed during user navigation, based on information such as title and code of each screen.

Similarly, in (El-Ramly, Stroulia, Sorenson, 2002) is presented another proposal of a method for legacy system reverse engineering based on systems execution logs mining. The difference between this proposal and the proposal presented above is that this one led to the discovery of use cases in the system. Below are listed the main features of the method: application of associative mining algorithm. Data mining phase, aiming patterns discover. The discovered patterns are used in the process of use cases identification, which serve as basis for mapping the various business rules embedded in source code. Once a pattern is discovered, it is incremented, with the aid of human experts.

3 BUSINESS RULES SET

In (Kalsing, Nascimento, Iochpe, Thom, Reichert, 2010a), we define a set of 8 business rules required for mapping legacy systems to business process models, as proposed by our rewriting method.

Now, in this work, for each business rule type, we define a syntactic structure that represents a code fragment which is necessary to characterize the implementation of the business rule in the legacy source code. We can use these structures as templates to identify the business rules in legacy source code.

For a better understanding of these rules, we subdivided our business rule set into two subsets:

1) Business rule for work units: These rules are atomic operations executed in legacy system. They represent units of work (processing) that are indivisible.

2) Business rule for control flows: These rules are operations that control the flow of execution of the legacy system. They represent conditional, loop and others structures of control flow.

In this paper it will be studied methods which allow identification of the following types of business rules (see Table 1).

The syntactic structures presented in Table 1 can be found in most imperative programming languages.

Table 1: Business Rules Classification.

Business Rule Type	Description
Business Rules for Work Units	
Math Calculation	Represents a math calculation. It can be a simple math expression or a complex calculus related to several statements of code. Ex: *Discount = (price*20)/100;*
Function or Procedure Call	Represents an external function or procedure call in source code. An external function or procedure is usually a black box, where the source code is not available. Ex: *ret = InvokeCreditAnalysis(customer);*
Data Persistence	Represents statements associated to data manipulation. It can be a SQL statement or a file I/O operation. Ex: *sql = "SELECT * FROM customer WHERE name = " + customer_name;*
User Interaction	Represents an input or output data form, where the user interacts with the system. Ex: *printf("*Customer Name/Address: *"); scanf("%s", customer_name); ...*
Business Rules for Control Flows	
Conditional Decision	Represents a decision, i.e., a conditional statement (e.g. if-then-else, switch, etc). Ex: *IF (approved && price>1000) { ... }*
Iteration	Represents a repetition, i.e., a loop statement (e.g. for, while, etc). Ex: *WHILE (items > 0) { ... }*
Exception Handling	Represents special structures to handle exceptions (e.g. try-catch). Ex: *try { ... } catch (Exception e) { ... }*
Parallel	Some programming languages can execute commands in parallel through the structure *fork/join*. Ex: *fork { ... } : { ... } join*

4 DESIRED FEATURES FOR BUSINESS RULES IDENTIFICATION METHODS

In this section are presented desired features of legacy systems reverse engineering methods obtained in bibliography research.

1) **Application Domain Independent:** second (Newcomb, Kotik, 1995), the method should be generic enough to not be directed to specific characteristics of the application domain where the analyzed system is operating;

2) **Programming Language Independent:** it would be interesting to have a method of legacy systems reverse engineering that is not tied only to systems implemented in a particular programming language (Kuipers, Moonen, 2000).

3) **Scalability:** second (Almonaies, Cordy, Dean, 2009) it would be interesting to have a method of systems reverse engineering which provide good scalability when applied to the whole legacy system,

or when applied to large modules. Therefore, it is expected that the method is capable of identifying the proper relationship between the various business rules contained in the modules of the system and also be able to manage successfully the question of the growing volume of data to be analyzed.

4) **Types of Business Rules Identified:** the method should be able to recognize at least the types of business rules listed in section 3.

5) **Reduce Human Intervention:** second (Paradauskas, Laurikaitidis, 2006) the search for an automatic legacy systems reverse engineering method is the main objective of a great number of papers available in literature. It would be great to be able to decrease the amount of human intervention in the process of business rules identification, either during the step of extracted data validation, either during the stage of business rules annotation in legacy systems source code, before mining the execution logs.

5 PROPOSAL OF A SUITABLE METHOD FOR BUSINESS RULES IDENTIFICATION

A hybrid method will be utilized to identify the whole set of business rules listed in section 3. That is, our method of business rules identification will use both source code manipulation and execution logs mining techniques.

The application of a reverse engineering method based on source code manipulation, using AST, dependency graphs, domain variables management and program slicing technique is suitable for the identification of business rules for work units. The application of techniques based on systems execution logs mining is suitable for the identification of business rules for control flows.

The implementation of our method consists of two algorithms: the first algorithm use techniques of source code manipulation to identify business rules for work units; the second algorithm use systems execution logs mining to identify business rules for control flows.

The first algorithm identifies the business rules for unit works and to annotate both the start and end points of the business rules found in the source code (Figure 1(a)). Annotations are source code lines that write records into a log file during next executions of the legacy code (Figure 1(b)).

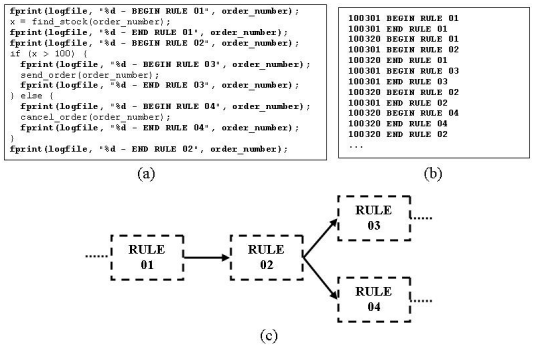

Figure 1: Main Steps of our Method.

Below is presented the steps of the algorithm to make the source code manipulation and to annotate the business rules for work units:

1) AST Creation: an AST representing the whole system is obtained after source code parsing (Putrycz, Kark, 2007) (Paradauskas, Laurikaitidis, 2006);

2) Generation of Dependency Graphs: the dependency graphs are obtained based on the AST. These graphs are used to assist the systems source code understanding task, representing the relationships between the different variables and instructions present in the source code (Wang, Sun, Yang, He, Maddineni, 2004a) (Horwitz, Reps, Binkley, 2003) (Ottenstein, Ellcey, 1992);

3) Domain Variables Identification: based on variables classification (Chen, Tsai, Joiner, Gandamaneni, Sun, 1994) and input/output variables management (Wang, Zhou, Chen, 2008);

4) Application of Program Slicing Technique: guided by the domain variables identified in the previous step (Xu, Qian, Zhang, Wu, 2005). Second (Harman, Hierons, 2001), static program slicing is more suited for business rules identification in source code, because it works representing all the possible branches that the systems execution flow can follow, unlike other program slicing approaches, such as dynamic, conditional and amorphous slicing;

5) Extracted Data Validation: in the end of the process, human intervention is needed, in order to validate the extracted business rules. The validation step is necessary because none of the methods available in literature is able to automatically identify the business rules ensuring an acceptable level of accuracy.

After the business rules for work units annotation, a set of meaningful execution scenarios is selected according to the most frequently used legacy system running parameters. In the sequence, the system is recompiled and executed repeatedly

according to the selected running scenarios. Thus, a log file will be created and extended with information about business rules executions within most probable running scenarios (Figure 1(b)).

In the second step, the system execution log file is analyzed by the process mining algorithm. This algorithm is called Incremental Miner and has been developed by the research group. The Incremental Miner Algorithm was presented in (Kalsing, Thom, Iochpe, 2010b). The Incremental Miner Algorithm analyses the log file and produces as its output one or more probable partial orders of business rules executions according with some predefined threshold (Figure 1(c)).

Note in Figure 1(c) that the boxes Rule 01, Rule 02, Rule 03 and Rule 04 represent the execution of the business rules for work units annotated by the algorithm of source code manipulation. Note also that there are two transitions out of the Rule 02. These transitions represent a business rule for control flow. Thus, the Incremental Miner algorithm captures the behavior of legacy source code control flow instructions, i.e., it identifies the business rules for control flows.

The combination of the source code manipulation and the systems execution logs mining incremental algorithm is able to recognize the set of business rules listed in section 3 and also provides the desired features for legacy systems reverse engineering methods listed in section 4.

6 SUMMARY AND OUTLOOK

In this paper were studied and analyzed methods of legacy systems reverse engineering, which allow knowledge extraction, emphasizing business rules identification and extraction.

Throughout the text, we presented its main characteristics and a comparison between the different techniques, data structures and concepts used by the cited methods.

The hybrid method, based on source code manipulation, which uses AST, dependency graphs, domain variables management, program slicing technique and systems execution logs mining, presented in section 5 is suitable to identify and extract business rules from legacy systems. In the end of the process, human intervention is needed to validate the extracted data. The validation step is necessary because none of the methods studied is able to automatically identify the business rules ensuring an acceptable level of accuracy.

It is on course the implementation of a prototype which will be able to apply to legacy systems the reverse engineering hybrid method presented in section 5. In future work, using the prototype and the knowledge obtained in this work, we intend to be able to identify and extract the whole set of business rules listed in section 3.

REFERENCES

Almonaies A., Cordy J., Dean T., Legacy System Evolution towards Service-Oriented Architecture, 2009.

Chiang C. Extracting Business Rules from Legacy Systems into Reusable Components, 2006.

El-Ramly M., Stroulia E., Sorenson P., Mining System-User Interaction Traces for Use Case Models, 2002.

Erlikh L., Leveraging legacy system dollars for e-business, *IT Professional*, p. 17-23, 2000.

Harman M., Hierons M., An overview of program slicing. *Software Focus* p. 85-92, 2001.

Huang H., Tsai W., Bhattacharya S., Chen X., Wang Y., Sun J., Business Rule Extraction from Legacy Code. In: *20th Compute software & Applications Conf. IEEE Computer Society Press* (1996).

Kalsing A. C., Nascimento G. S. do., Iochpe C., Thom L. H., Reichert M. An Incremental Process Mining Approach to Extract Knowledge from Legacy Systems, In: *14th Int. IEEE EDOC Conf., IEEE Computer Society Press*: Vitória-Brazil, (2010).

Kalsing A. C., Thom L. H., C. Iochpe. An Incremental Process Mining Algorithm. In: *12th Int. Conf. on Enterprise Information Systems*, (2010).

Knolmayer G., Endl R., Pfahrer M.. Modelling Processes and Workflows by Business Rules, In: LNCS - Business Process Management, *Springer: London*, (2000).

Kuipers T., Moonen L., Types and Concept Analysis for Legacy Systems, 2000.

Nascimento G. S., Iochpe C., Thom L. H., Reichert M. A Method for Rewriting Legacy Systems using Business Process Management Technology, In: *11th Int. Conf. on Enterprise Information Systems*, (2009).

Newcomb P., Kotik G. Reengineering Procedural Into Object-Oriented Systems, 1995.

Paradauskas B., Laurikaitidis A., Business Knowledge Extraction from Legacy Information Systems, *Information Technology and Control*, Vol.35, No.3, 2006.

Pressman R. S. Software Engineering: A Practitioner´s Approach, (*McGraw-Hill*, 2001).

Putrycz E., Kark A. W., Recovering Business Rules from Legacy Source Code for System Modernization, *Lecture notes in computer science*, p 107, 2007.

Ross R. G. The Business Rule Book: Classifying, Defining and Modeling Rules, *2nd edition, Business Rule Solutions*, (1997).

Stroulia E., El-Ramly M., Kong L., Sorenson P., Matichuk B., Reverse Engineering Legacy Interfaces: An Interaction-Driven Approach. *In Proc. of the 6th Working Conf. on Reverse Engineering*, (1999).

Van der aalst W., Reijers H., Weijters A., Business Process Mining: An Industrial Application, (2007).

Wang C., Zhou Y., Chen J. Extracting Prime Business Rules from large legacy system, *Int. Conf. on Computer Science and Software Engineering*, (2008).

Wang X., Sun J., Yang X., He Z., Maddineni S. Business Rules Extraction from Large Legacy Systems, In: 8th Euro. *Conf. on Software Maintenance and Reengineering*, 2004.

Weske M. Business Process Management: Concepts, Languages, Architectures, *Springer*; Berlin (2007).