A METHOD FOR REWRITING LEGACY SYSTEMS USING BUSINESS PROCESS MANAGEMENT TECHNOLOGY

Gleison Samuel do Nascimento, Cirano Iochpe
Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil

Lucinia Heloisa Thom, Manfred Reichert
Institute of Databases and Information Systems, University of Ulm Oberer Eselsberg, 89069, Ulm, Germany

Keywords: Legacy system, Reenginering, Business process, Business process management.

Abstract: Legacy systems are systems which execute useful tasks for the organization. Unfortunately, to maintain a legacy system running is a complex and costly task. Thus, in recent years several approaches were suggested to rewrite legacy systems using contemporary technologies. In this paper we present a method for rewriting legacy systems based on *Business Process Management (BPM)*. The use of BPM for migrating legacy systems facilitates the monitoring and continuous improvement of the information systems existing in the organization.

1 INTRODUCTION

Legacy systems are information systems which execute useful tasks for an organization, but were developed with technologies no longer in use (Ward and Bennett, 1995). Legacy systems include information and procedures which are fundamental for the operation of the organization. However, to maintain a legacy system running is a complex and costly task. Reason is that the corresponding program code can be obsolete, hard to understand, and poorly documented.

To reduce these problems, many organizations are opting for rewriting and implementing their systems using contemporary technologies. In recent years several approaches were suggested to perform this rewriting. The majority of them either requires the rewriting of the entire system (Biggerstaff, 1989) or by specific system modules (Bianchi et al., 2003). Both approaches block maintenance of the legacy system during the rewriting process. However, as business in organizations is generally dynamic and interruption of maintenance activities might result in operational problems. Apart from this, none of the two approaches consider the need of understanding how the legacy system works (i.e. what its business logics) and what its impact on the efficiency of the organization's business is.

Related to these problems, in this paper we present a method for rewriting legacy systems based on *Business Process Management (BPM)*. During the last years we have seen an increasing adoption of BPM tools by enterprises as well as emerging standards for business process specification, execution and monitoring.

Our method uses BPM and SOA in order to realize the rewriting of legacy systems. Basically, we aim at: 1) identifying business processes embedded in a legacy system and, 2) implementing the businesses process identified from legacy code using BPM and Service Oriented Architecture (SOA) tools, without need of rewriting the source code of the legacy system. The major contributions of our approach can be summarized as follows:

1. The business process being executed in the organization can be better documented, i.e., the business model obtained from analyzing the legacy code can be represented graphically in BPM tools.
2. The business process is explicitly represented in an executable model, which eases monitoring and improvement.
3. Fragments of the source code existing in the organization can be reused; i.e., the functions it comprises can be transformed in web services which are composed using a BPM tool.
4. Finally, after discovering the business process be identified our approach allows to perform the rewriting of the legacy code for each activity, i.e.,

Samuel do Nascimento G., Iochpe C., Heloisa Thom L. and Reichert M. (2009).
A METHOD FOR REWRITING LEGACY SYSTEMS USING BUSINESS PROCESS MANAGEMENT TECHNOLOGY.
In *Proceedings of the 11th International Conference on Enterprise Information Systems - Information Systems Analysis and Specification*, pages 57-62
DOI: 10.5220/0001950400570062

only the code (e.g., in Cobol) related to a particular business process activity will be rewritten (e.g., in Java).

The remainder of this paper is organized as follows: Section 2 gives background information needed for understanding this paper. Section 3 presents the proposed method for legacy system rewriting based on BPM. Section 4 discusses related work. We conclude with a summary and outlook in Section 5.

2 BACKGROUND INFORMATION

An organization is composed by business processes. Each process consists of a series of (structured) activities which jointly realize a particular business goal (Weske, 2007). For instance, the creation of a sale order in the organization can be seen as a business process where the sale of a product is the goal to be achieved. In order to achieve this goal a number of activities need to be performed such as stock checking, payment terms definition, and credit checking.

This holistic approach on enterprises, where business processes are the main instrument for organizing their operations, is called *BPM (Business Process Management)* (Smith and Fingar, 2002). BPM defines a life cycle to develop, implement, enact and monitor business processes (Weber et al., 2009). Resumidamente, the cycle has four phases: *Design Configuration*, *Enactment* and *Evaluation Phase*.

In the *Design Phase*, the business process to be executed in the organization is identified, its goals are defined and the respective process model is designed. In the *Configuration and Enactment Phases* respectively, the process model is implemented and executed. Finally, in the *Evaluation Phase* the process is monitored and diagnostics on its efficiency are obtained.

One of the major goals of BPM is to gain a better understanding of the operations a company performs and of their relationships. The explicit representation of business processes constitutes the core concept to achieve this better understanding. BPM also facilitates business process improvement. Currently, there are several tools to support each phase of the BPM lifecycle. These tools are called *Business Process Management System* (Reis, 2007).

In organizations, however, there exist information systems which were developed before BPM technology have emerged. These legacy systems do not give a clear perspective of the business process. Generally, they implement the complete business process, or fragments of it. There is a need to integrate legacy systems with current BPM approaches.

Legacy systems are typically complex systems which were developed using different programing languages. In this context *Service Oriented Architectures* have proven to be as efficient technology (Papazoglou and Heuvel, 2007). Web Services are interfaces that ensure the communication of systems developed with different technologies (Papazoglou and Heuvel, 2007).

The joint use of both BPM and SOA can be efficient in order to realize the rewriting of legacy systems. While BPM provides tools for building and implementing business processes, SOA provides a standard interface that allows to connect business processes and legacy systems through *Web services* generated from the source code of the legacy system.

3 PROPOSED METHOD TO LEGACY SYSTEMS REWRITING

In this section we introduce a BPM driven method we are developing to rewrite legacy systems. Our method comprises three phases: *Business Process Identification and Definition; Business Process Implementation; Business Process Enactment and Improvement*.

Note that a legacy system can implement more than one business process. Therefore, we are not necessarily proposing the mapping of the legacy system to one single business process. Similarly, a legacy system may implement only fragments of a business process.

Thus, it is very important that the *developer* knows the business processes that the legacy system implements (*developer* is the user that is applying the proposed method this paper). Accordingly, the developer must apply our rewritten method to each of the embedded processes in legacy system. Note that in each iteraction of our method only one business process is identified and rewritten. The legacy system then is rewritten by modules, where each module corresponds to a business process. Therefore, implications related to the legacy system maintenance are minimized.

In the following sections we describe each of these phases in detail.

3.1 Business Process Identification and Definition

To identify business process from legacy code and to define respective models constitute the main phase of the proposed method, It is in this phase in which the

developer designs the business process. For this purpose, he identifies from the legacy system the process activities to be implemented.

Many steps of this phase require human intervention. However, in particular steps of our mapping approach we can use algorithms which shall help the developer in extracting the desired information from the legacy code.

In order to better understand this phase we subdivide it in eight steps executed in sequence.

Step 1: Scope Definition of the Process. In Step 1 the developer must define the purpose of the business process within the organization and its application domain. In addition, the developer must list keywords related to the defined goals and application domain. As example, let us assume that the organization has a legacy system to monitor sales. Assume further that we want to know how a quotation of sale is performed in the organization. Then, the developer first has to define the process purpose: *generation of a quotation*. Second, the application domain of the process must be identified. As it is executed in the sales department, the application domain is *logistics*. Finally, keywords related to the process must be defined (e.g. *product, stock, customer, credit*).

The definitions made in this step can be used in the development of algorithms to extract information from the source code in the subsequent steps. Through keywords, such as *stock*, an algorithm executed in Step 7, *(Identification of Automatic Activities)*, can find a procedure named "stockControl", and automatically map it to an automated activity of the process. Note that the activity is automatic, since it is executed for a procedure (or function) of the legacy system.

Step 2: Defying of Start and End Events. At this step, the developer must identify the start and end events of the process. The start event may be the display of a screen of the legacy system, or even the receipt of a file or message in the organization. Considering the quotation example, for instance, the start event of a quotation may be a *system operator* accessing the menu *Quotation* and selecting the option *Create a New Quotation*. Similarly, the developer must identify the event or condition that terminates the business process. In our example, the end event may be the recording of the order in a database, or even the *system operator* receiving a confirmation message on the screen. Obviously, a business process can have more than one start or end event. Through start and end events, we can select the source code files to be analyzed in order to identify the business process.

Step 3: Identifying Human Activities in the Legacy System. In Step 3 the developer must indicate the human interactions with the legacy system. The human interactions considered here, correspond to the electronic forms of the legacy system being involved between the start and end events (cf. Step 2). Obviously, these forms correspond to human activities in the business process.

In the quotation example, the forms are filled out by an operator of the legacy system during the creation of a quotation (e.g. the form of client information). Each human interaction is mapped to a human activity in the business process. Observe that in this step the fields comprised by each form can be identified. This information can be used in the creation of metadata which can be further used in the generation of forms related to the human activities of the business process.

The metadata can be also useful when identifying in the source code the automated activities related to the business process. In the source code we can find business rules which validate the values entered in the fields of the forms, as we show in Step 7.

Step 4: Identifying Activities Outside the Legacy System. As before mentioned, a legacy system does not always covers all activities of a business process. Eventually, the process may also have activities which are not executed within the scope of the legacy system. An example of this kind of activity in the quotation process is the credit verification of a customer within an institution of credit protection. Other examples include the production of items that are not in stock and the shipment of items to the customer. In addition to the identification of these activities, the developer should indicate whether the activity is manually or automatically executed.

Step 5: Defining the Partial Order of the Activities Identified in Steps 3 and 4. After identifying the human activities implemented in the legacy system as well as the activities which are not executed within the scope of the legacy system, the developer must define the partial order of these activities in the process model. The definition of the partial order is done manually by the developer.

Obviously, these activities are connected by specific control flows (e.g., sequence, XOR-Split, AND-Split). The developer must indicate control dependencies between the activities.

The partial order of human activities can also determine which part of the source code is analyzed in Step 7. When determining the relationship between human activities A and B the developer is delimiting the portion of source code to be reviewed when identifying automated activities between A and B.

Step 6: Identifying Roles for Executing Human Activities. At this point, the developer has already identified the human interactions within the legacy system, and has defined partial order between these activities. Now, he must verify which are the roles or responsibilities in the legacy system, that a user must have to fill each of the forms identified as human activities (Ly et al., 2005). This allows to identify user roles within the business process require to work on respective activities.

Step 7: Identifying Automated Activities. In Step 7, the developer must analyze the source code in order to identify automated activities being executed in the legacy system. For that, the partial order of human activities is taken into consideration. Like for Step 5, the identification of automated activities should consider the source code being executed between the invocation of two human activities of the business process.

Afterwards, the developer must identify the business rules captured by this source code. A business rule is a statement that control or influence the behavior of a system (Group, 2006).

Regarding the source code of a system, business rules are structures of type *condition* and *action*; a *condition* consists of one or more *boolean* expressions connected by logic operators (*"and", "or", "not"*). An *action*, in turn, consists of operations which produce some processing result (e.g., recording of data in a database).

Below we list characteristic business rules we are trying to discover from the source code of the legacy systems.

- **Persistence**: Code fragments which deal with data persistence. Usually, these fragments refer to database transactions.
- **Information Flow**: Rules defining the information being exchanged between two activities.
- **Control Flow**: Rules defining the routing of automated activities (e.g. *IF* statements).
- **Pre-conditions** Rules indicating required conditions which must be satisfied in order to execute a particular activity. For example, after performing a human activity, there be validations in the fields.
- **Pos-conditions**: Rules verifying a system condition after executing a determined data processing.
- **Frequency**: These rules identify the number of iterations of a particular rule or a series of rules within the source code (i.e. *loops* in source code).
- **Execution Time (Duration)**: Business rules which define the execution time of an activity. Fragments of the source code that identify *time-outs* of processing.
- **External Calls**: These rules determine the invocation of external applications from the legacy system.
- **Computations**: Rules which identify mathematical computations.

Here we can develop algorithms to support the developer in the identification of automated activities. Currently, there exist several algorithms for identifying business rules in source code (Tip, 1995) (Paradauskas and Laurikaitis, 2006). We intend to adopt these algorithms in order to identify the most relevant business rules of a particular business process. Note that these algorithms must consider all information collected during Steps 1 to 6.

Step 8: Validation of the Retrieved Process Model. After having identified automated activities, as well, we obtain, the final process model. The obtained process model has to be validated. Here, behavior properties of the process model are verified (e.g., absence of *deadlock*); there are paths that are never executed. In order to perform this validation, the business process is mapped to a representation in π-calculus (Milner et al., 1992). This representation is called π-process. The π-process obtained can then be checked with a model ckecking tool. In (Thom et al., 2008) this mapping is defined and a model checking tool is detailed.

3.2 Business Process Implementation

The implementation phase of the process model is divided into three steps. Note that the implementation of process model must avoid operational breaks in the legacy system, i.e., the legacy system must continue operating and its maintenance must not be interrupted.

Step 1: Implementation of the Process Model. In the first step, the developer must choose a BPM tool for implementing the business process (e.g., Intalio (Intalio, 1999) or ADEPT2 (Reichert et al., 2005)). After that, the process must be designed in a notation supported by the tool. Moreover, the developer must define input and output attributes of each activity in the process as well as the related roles and users which execute them.

In addition, the developer may want to implement procedures to obtain measurement metrics on process performance.

Step 2: Implementation of Automated Activities. We can now implement the automated activities of the process model. As discussed in Section 3.1 (Step 7), these activities are identified based on the analysis of the business rules existent in the source code. Through business rules identification we can also

identify the source code implementing these rules. This source code can be mapped to automated activities of the process model.

We can implement a web service to execute the piece of source code which implements an automated activity. Thus, it is not necessary to rewrite the source code in a language that can be interpreted by the BPM tool. For example, suppose that we are using a legacy system written in *C* language. In this case, we can write a Java webservice, which will call a procedure in *C* language, through *JNI (Java Native Interface)* (Liang, 2002). The Java code generated is a method that calls the *C* procedure.

Altogether, our approach allows for the migration of legacy systems to a process oriented technology without need of rewriting the legacy code, and without interrupting the operation or maintenance of the legacy system.

After creating the web services, they must be connected to the respective activity in the business process implemented in the BPM tool.

Step 3: Implementation of the Human Activities. In the final step of the implementation phase, the forms related to the human activities are generated. The attributes included in each form are the same appearing in the the screens of the legacy system.

In the most majority of BPM tools, forms are automatically generated. Giving the attributes related to each process activity and respective data types, the tools generate the forms. The form can then be customized according to a particular domain.

3.3 Business Process Enactment and Improvement

In the last phase of our method the process model is executed and the results its execution monitored. Based on the analysis of the results it is possible to improve the process model as well as the performance of each activity.

In this step, the organization also may consider to rewrite the source code in a more contemporary language. In the example given in Section 3.2, Step 2, we use JNI to invoke a procedure written *C*, thus allowing the reuse of legacy code. After implementation and validation of the business process, the developer may begin to migrate from *C* code for a language as *Java*. Thus it eliminates the *JNI* communication in the future. This change can be executed without any reflection on the operation of the organization.

4 RELATED WORK

In recent years, several methods for legacy system rewriting have been proposed and discussed in literature. We classify these methods into three groups: 1) reengineering of the complete system; 2) reengineering through *wrapping* techniques; 3) reengineering of the system by modules.

An example of the first category include the approach proposed in (Biggerstaff, 1989). These methods consider the migration of the complete system. This implies the locking of maintenance in legacy system, in order to prevent whatever change in the organization business rules during the rewriting process. In our proposal this does not occur for two reasons: First, the system is not completely rewritten, but rather the parts related to the business process. The second reason is that the source code of the legacy system is not rewritten in a first moment, as shown in Step 2 of the implementation phase of the business process. The code is rewritten after the process be running, as proposed in Section 3.3.

The second category, reengineering using *wrapping* techniques, proposes the introduction of a communication layer between the new system and the libraries of the legacy system, as proposed in Step 2 of the implementation phase of the business process. The approach proposed in (Bisbal et al., 1999) is a sample of the respective techniques. However, these studies do not consider analysis of the legacy code. They propose that only new features of the system are using a new technology, and legacy code is transformed into a black box i.e., they do not provide an explicit view of the business process. In our proposal, the business process is documented in a BPM tool, so it can be executed, monitored and improved.Finally, the third category considers reengineering by modules. In this case, the legacy system is divided into modules. Thus, during maintenance only the involved modules are locked rather than the complete system. The methodology *Iterative* (Bianchi et al., 2003) is example of this category. Our proposal differs from this category because the main focus is on the business process being executed in the organization. Moreover, the identification of the modules of the legacy system is complex. In our approach, we propose the identification of the business process enabling then the identification of the related source code.

5 SUMMARY AND OUTLOOK

In this paper we proposed a BPM driven rewriting approach which consists of the mapping of legacy systems to a business processes. The overall goal of our rewriting method is to ease the upgrade of the legacy system. The use of BPM for migrating legacy systems facilitates the monitoring and continuous improvement of the information systems existing in the organization. In addition, the business process being executed in the organization is documented. This, in turn allows for dissemination of the knowledge, which was previously only by the developers of the legacy system.

Another significant advantage of our rewriting method concerns in the head of the reuse of the source code of the legacy system. Thus, the business process can be implemented without the legacy system being interrupted or its maintenance being blocked. After the process be running, the organization can start the rebuilding of legacy code and the business process will not be interrupted. Furthermore, we have already started to use our method to the rewriting of real legacy system from the logistic domain.

As future work we intend to develop data structures, which shall store and integrate all the concepts discussed during the application of our method. For instance, the construction of an ontology, which relates to business rules, processes activities and application domain. These structures are fundamental for building algorithms.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the Coordination for the Improvement of Graduated Students (CAPES), the Institute of Databases and Information Systems of the University of Ulm (Germany), and the Informatics Institute of Federal University of Rio Grande do Sul (Brazil).

REFERENCES

Bianchi, A., Caivano, D., Marengo, V., and Visaggio, G. (2003). Iterative reengineering of legacy systems. *IEEE Transactions on Software Engineering*, 29(3):225–241.

Biggerstaff, T. J. (1989). Design recovery for maintenance and reuse. *Computer*, 22(7):36–49.

Bisbal, J., Lawless, D., Wu, B., and Grimson, J. (1999). Legacy information systems: Issues and directions. *IEEE Software*, 16(5):103–111.

Group, B. R. (2006). Guide: Business rules project. Technical report. Disponível em: *www.guide.org/pubs.htm*.

Intalio (1999). Creating process flows. Technical report, Intalio Inc.

Liang, S. (2002). *Java Native Interface: Programmer's Guide and Specification*. Sun Microsystems, Inc.

Ly, L. T., Rinderle, S., and Reichert, M. (2005). Mining staff assignment rules from event-based data. In *In: Proc. Workshop on Business Process Intelligence (BPI) in conjunction with (BPM'05)*, pages 177–190, Nancy, France. Springer.

Milner, R., Parrow, J., and D., W. (1992). A calculus of mobile processes. Technical report, University of Edinburgh.

Papazoglou, M. P. and Heuvel, W.-J. (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal-The International Journal on Very Large Data Bases*, 16(3):389–415.

Paradauskas, B. and Laurikaitis, A. (2006). Business knowledge extraction from legacy information systems. *Information Technology and Control*, 35(3):214–221.

Reichert, M., Rinderle, S., Kreher, U., and Dadam, P. (2005). Adaptive process management with adept2. In *ICDE '05: Proc. Int. Conf. on Data Engineering*, pages 1113–1114, Tokyo, Japan. IEEE Comp. Press.

Reis, G. (2007). Introduction to bpm, bpms and soa. *Portal BPM*, 01:22–29.

Smith, H. and Fingar, P. (2002). *Business Process Management: The Third Wave*. Meghan-Kiffer Press.

Thom, L. H., Iochpe, C., Reichert, M., Weber, B., Matthias, D., Nascimento, G. S., and Chiao, C. M. (2008). On the support of activity patterns in prowap: Case studies, formal semantics, tool support. *Revista Brasileira de Sistemas de Informacao (iSys)*, 01.

Tip, F. (1995). A survey of program slicing techniques. *Journal of Programming Languages*, 3:121–189.

Ward, M. P. and Bennett, K. H. (1995). Formal methods for legacy systems. *Journal of Software Maintenance and Evolution*, 7(3):203–219.

Weber, B., Reichert, M., Wild, W., and Rinderle-Ma, S. (2009). Providing integrated life cycle support in process-aware information systems. *Journal of Cooperative Information Systems*, 18(1). (Accepted for Publication).

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