A business process metamodel for Enterprise Information Systems automatic generation

Luiz Fernando Batista Loja[1], **Valdemar Vicente Graciano Neto**[1], **Sofia Larissa da Costa**[1], **Juliano Lopes de Oliveira**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
P.O. Box 131 – Campus II – CEP 74001-970 – Goiânia – GO – Brazil

{luizloja, valdemar, sofia, juliano}@inf.ufg.br

Abstract. *Business Processes Modeling (BPM) is a critical success factor for Enterprise Information Systems (EIS), since a process represents the know-how accumulated by the organization and improves the productivity of its members. Model-Driven Development (MDD) approaches are being increasingly applied to generate Information Systems software, but current proposals have failed in observing BPM aspects and, therefore, they do not support many requirements of business processes execution and management. This paper proposes a business process metamodel to assist MDD initiatives in providing such capabilities to the generated Information System software. The proposed metamodel adapts and extends the BPMN [OMG 2009] concepts in order to build the modeling primitives to MDD initiatives. This metamodel was applied to improve an existing MDD framework that generates and manages code for EIS.*
Keywords: *Business processes metamodel, model-driven development.*

1. Introduction

In modern organizations, Business Processes (BP) are extensive, complex, dynamic, long-standing and can be defined as mutually dependent activities which produce a specific service or product [Smith and Fingar 2002]. BP Modeling (BPM) is a key instrument to organize BP, improving the understanding of the activities and their interrelationships [Wesk 2007]. Information Systems (IS) are the main tool to support and control the execution of BP. Although with current technologies, maintaining IS is an increasingly difficult task.

Model-Driven Development (MDD) is gaining prestige from both academic and corporative communities as an efficient approach to the maintenance and evolution of IS. This popularity is increasing because MDD offers the potential for automatic transformation of high-level abstract models into running systems [Mellor et al. 2003]. For instance, the model-based framework described in [Almeida et al. 2009] allows generating automatically a large portion of an EIS software, based on the conceptual model of its domain. The framework adopts a metamodel that conceptually represents the structural aspects of an EIS domain, such as business concepts, relations and static constraints. This metamodel is an object-oriented variation of the classic Entity-Relationship (ER) Model and can be considered a Domain-Specific Language (DSL) for EIS domains.

Although the framework [Almeida et al. 2009] provides MDD facilities to manage the automatic generation of business rules, database evolution, and graphic user

interface code, it does not support business processes definition and execution. Other limitations of this framework are: user interface and business domain features are tightly coupled in its metamodel; the generated user interface is not customizable; and it does not support BPM.

In order to overcome these limitations, an integrated refactoring effort is being conducted to evolve this framework. The refactoring activities resulted in three improvement aspects: **Business Domain**, **User Interface** and **Business Process**. These aspects are the three main areas of concern when building EIS, and are the cornerstone of the new framework architecture. A metamodel is being developed for each one of these aspects and the framework provides the integration and interoperation of the three aspects.

This paper describes the Business Process Metamodel of the improved framework. Based on this metamodel, a BP management component is integrated to the framework, extending its functionalities to support BPM and the automatic generation and execution of code for the modeled processes. Due to space limitations, the Business Domain and User Interface metamodels are not discussed in this paper, which is structured as follows: Section 2 justifies our proposal; Section 3 introduces the BP Metamodel; Section 4 describes the process execution engine developed to support the MDD approach for BP within the framework; Section 5 discusses an example of application of the improved framework; and Section 6 concludes the paper analysing related work.

2. MDD and BPM

Many frameworks have been recently proposed to support MDD. [Achilleos et al. 2007] defines a metamodel as a complete and precise specification of a Domain-Specific Modelling Language (DSML), which in turn can be used to define models of a domain. Thus, a metamodel is one abstraction layer higher than domain models. This paper proposes a metamodel to contemplate BP domain specification within the EIS domain.

Although there is a large number of notations, languagens and tools for BPM, the Business Process Modeling Notation (BPMN) is becoming the *de facto* standard [OMG 2009]. BPMN is a process modelling notation and a process specification language allowing to capture and exchange BP models among stakeholders and tools [Großkopf 2007]. However, BPMN has some drawbacks that justifies the creation of a DSML instead of using the general purpose notation in MDD initiatives [Nuffel et al. 2009] such as: 1) only a subset of BPMN constructs is actually used when building BP diagrams [zur Muehlen and Recker 2008]; 2) a number of BPMN artifacts have no clear real-world meaning (for instance the Text Annotation construct); 3) BPMN lanes and pools are redundant, because the same thing can be modeled using a pool or a lane; 4) BPMN is not suitable to model business rules, data and information models; 5) BPMN lacks unambiguous definition of the notation.

These limitations of BPMN lead us to propose a new BP Metamodel specifically adapted to assist EIS process modeling. This metamodel, specified in the next section, presents a reduced set (but not necessarily minimal subset) of BPMN constructions capable of representing BP within EIS, diminishing the redundacy, eliminating some explicit ambiguities, and integrating, via MDD, the BP with major EIS features such as business rules, business data and user interfaces.

3. The Business Process Metamodel

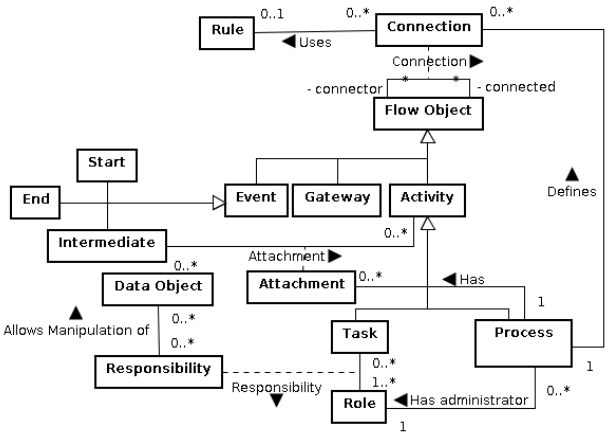


Figure 1. The Business Process Metamodel

The main concept of BP Metamodel is the *Process* (Figure 1). A Process is a well described activity which defines Connections and has Attachments linked to Activities. A *Connection* is responsible for linking two *Flow Objects*, the *connector* and the *connected* object. A Flow Object can be a *Gateway*, an *Event* or an *Activity*. Process execution flow goes from connector to connected flow object. An *Attachment* defines process boundary events and is linked to an activity. Events can be of three types: Start, Intermediate and End. Gateways will determine branching, forking, merging, and joining of paths.

An *Activity* can be atomic (*Task*) or non-atomic (*Process*). A *Task* is an atomic unit of work that can be performed by one person or a group. A Responsibility emerges from the association between a Task and a *Role*. Roles define what kinds of participants are necessary to accomplish the task. For each association between a Task and a Role, there is a Responsibility. This entity defines the role and the interaction type that a participant will assume when interacting with the task.

Concepts of Task, Role and Responsibility are defined according to the COBIT [ITGI 2009] model, in which a task may involve one or more participants with different types of interaction. While BPMN pools and lanes only define *who* do the task, COBIT states that there are four types of interactions: *responsible, accountable, consulted, informed*. These are not roles, but ways of interacting with a task. In our metamodel a task can be performed by many participants. Each one can interact in different ways with the same task. Aiming to support this type of responsibility definition, pools and lanes were discarded from the metamodel.

For each responsability, zero or more *Data Objects* can be associated. This relationship defines which objects will be allowed to be manipulated by the participants while executing the task. The concepts that connect the Business Process Metamodel, the User Interface and the Business Domain metamodels, which control the *Data Objects*, will not be detailed in this paper due to space limitations.

One interesting simplification of this BP metamodel with relation to BPMN is the elimination of the *Sub-Process* concept. In the metamodel, a process defines connections, which link flow objects that can be specialized in a process. Hence, a process can contain

other processes. This was done because, in BPMN, process and sub-process are almost the same concept. On the other hand, we consider that if there is something within a process, it is a task or a process. We define a process inside a process. This promotes the reuse of process definitions. If a sub-process concept is used, it is possible to infer that the sub-process is not an independent entity and can not be executed without the process that contains it. Thus, if only process and task concepts are provided in the metamodel, a process can be reused as an internal process by many processes - including itself (a recursive idea) - and can be executed independently. Moreover, supressing the concept of sub-process brought a recursive capability to the metamodel, supporting the recursive Process Pattern described by [Russell et al. 2006].

In addition, no Flow Object is defined within a process context. They have their own independent existence within the metamodel. This means that when a process is being modeled, Flow Objects are joined to each other through connections that belongs to a specific process. The primordial advantage is that a Flow Object can be reused in many processes. In addition, changes in the Flow Object will be reflected to all processes that share this object.

Aiming to support others Process Patterns described in [Russell et al. 2006] an *Administrator* concept was added to the metamodel. For each process there is one role that defines which kind of participant will be allowed to manage the process. This participant will be responsible for defining the *ad hoc* process flow and will be able to create instances of an activity when the process is a multiple-instance activity.

In order to eliminate redundancy, the Message Flow concept was eliminated from the metamodel, since it is rarely used by BPM users [zur Muehlen and Recker 2008]. The Association Concept was also removed from the metamodel because artifacts (a concept supported by BPMN) were not adopted in this metamodel.

Although Rules are created independently, they establish a relation with gateways that use them to control the flow.

The metamodel supports most of the Process Patterns defined in [van der Aalst et al. 2003] and other patterns described in [Russell et al. 2006]. According to [Russell et al. 2006] these patterns delineate the fundamental requirements that arise during BPM. The implementation of all these patterns grants considerably expressive power on BPM. Moreover, the MDD approach allows the automatic code generation for the modeled process, and the management of the runtime behavior of these processes. Therefore, it is possible to conclude that the main advantages of our metamodel are: 1) Focus on BPM reuse; 2) Reduction of the size of the model (following recommendations of [Nuffel et al. 2009, zur Muehlen and Recker 2008, zur Muehlen et al. 2007]); 3) Elimination of ambiguities and redundancies, such as pools and lanes; 4) Introduction of a comprehensive representation of roles involved in BP execution.

4. Process Execution Engine

To support the MDD approach on modeled processes, we created a BP Execution Engine (BPEE). Figure 2 shows the execution engine model that is capable to manage the execution of processes modeled with our metamodel. Gray blocks are related to the BPEE while white blocks are elements related to the BP Metamodel.

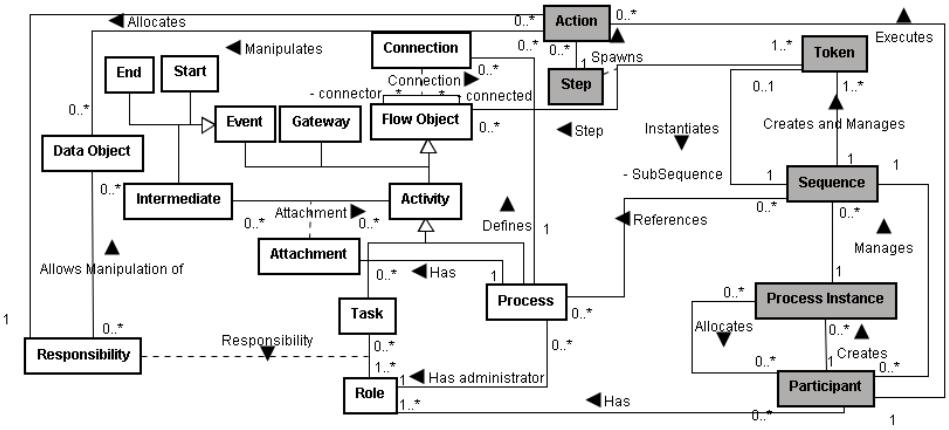


Figure 2. BP Execution Engine Model

After defining a process using the BP Metamodel is possible to instantiate the process, creating a *Process Instance*. This is done by giving the instance a name, and determining its *Participants* and their *Roles*. The Process Instance will manage the *Sequences* to perform the process. Thus, for each Process Instance an initial sequence is originated. This sequence references the process that was instantiated.

Sequences are responsible for creating and managing *Tokens*. A Token is used to mark which Flow Object is being executed at a given moment. For every Flow Object reached by a token, a *Step* is created with the token arriving timestamp. The Step marks when the Token reached the Flow Object and when it leaves this object.

When the token arrives at a task, the participants that were included at the process instance are allocated to accomplish this task. To determine the participants that will be allocated the BPEE analyses the roles that the process instance's participants can assume and the roles that the task requires.

When a token reaches a process a new sequence is created and initiated by this token. The token will wait until the sequence that was started is finished. When the sequence finishes the token will follow to the next Flow Object.

When a token reaches an event or a gateway it acts depending on the Flow Object that was reached. These behaviours are all based in the BPMN specification [OMG 2009].

After executing the behavior of the Flow Object, the token will continue to the next Flow Object until it reaches an End Event. When this happens, the token will be consumed by the event. This means that this token will not create more steps and its life cycle is over. A sequence only finishes when all tokens that it spawned have been consumed; a Process Instance only finishes when all its sequences have been finished.

The BPEE implements the MDD approach to support BPM and automatic generation of EIS BP code. The main advantages of this BPEE are: 1) it allows empirical process execution (i.e. it is possible to start a process with just one flow object - a start event, for example - improving the flexibility of the BPM); 2) it is possible to model and change a process during its execution; 3) it supports the Process Patterns mentioned in [van der Aalst et al. 2003], including *ad hoc* modality.

5. Metamodel Application: an Example

The BPEE supporting the BP Metamodel was implemented in a prototype, including Process Modeling and Execution functionalities. Figure 3 illustrates a process modeled using this prototype for a Web platform. This is also a differential feature of this work since, to the best of the authors knowledge, there is no tool that performs the whole process - modeling and execution - of BP in the web environment.

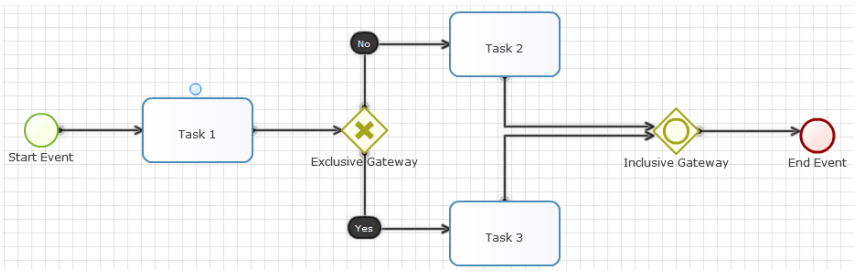


Figure 3. Example of Modeled Process.

Figure 3 describes an arbitrary process created to demonstrate some concepts of the BP Metamodel. The process begins with an start event that follows to task 1. This task is connected with an exclusive gateway that changes the flow according with connection rule. The flows that uses the rule "Yes" leads to task 3 while the connection which uses the rule "No" flows through task 2. Both tasks are connected to the inclusive gateway. This gateway will synchronize the flow and send the token to the End Event where it will be consumed. Rules will not be detailed here due to space limitations.

Figure 4 displays the object diagram of this simple process using the metamodel.

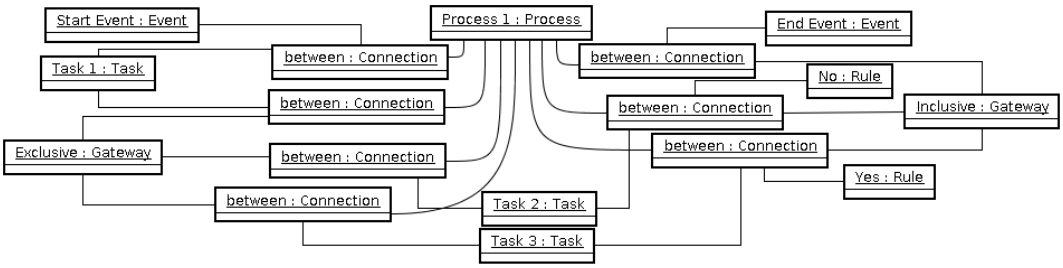


Figure 4. Sample Process Object Diagram

Start Event is one instance of a specific kind of Event (Start) defined in the metamodel. The ball above Task 1 represents the token. Task 1, Task 2 and Task 3 are metamodel tasks. Inclusive Gateway and Exclusive Gateway are metamodel gateway samples. And, finally, a final event (end event, in metamodel), finishes the process execution.

All Flow Objects are linked by Connection objects. Rules "Yes" and "No" are also linked to a Connection, which means that the token will flow through these connections only when their Rule expression return true.

6. Conclusions

This paper presented a Business Process Metamodel and a Business Process Execution Engine created to add BPM support capabilities to a new version of the framework de-

scribed in [Almeida et al. 2009]. These proposals will be integrated to other two initiatives to constitute a model-based framework capable of generating EIS with well-defined business processes.

The contributions of this work can be seen under two perspectives: the Business Process (BP) Metamodel and the corresponding BP Execution Engine. Considering this engine, there are many tools that are also designed to manage BP, such as Bizagi [Bizagi 2010], Bonita [Bonita 2010] and Enterprise Architect [Systems 2010].

These are Business Process Modeling Suites solutions that allow process modeling and execution using a graphical interface. These applications have modules that help users to optimize process without the intervention of a programmer, and also have form makers and others features that helps in process management. Although they let users modify process while they are running, this modification takes place only in the next process execution. In addition, the application does not let users execute incomplete processes, what turns the execution of empirical processes harder. Besides that, it is not possible to implement some patterns like the ad hoc pattern and the recursion pattern cited in [Russell et al. 2006]. Moreover the customers can't reuse already defined elements like events, gateways and process within other processes.

There are some papers which propose others metamodels extended from BPMN like: [Tsai et al. 2007], [Awad et al. 2009] and [Korherr and List 2006]. [Tsai et al. 2007] presents BPMN* that is a BPMN metamodel with some constraints. These constraints aim to solve BPMNs ambiguities, however BPMN* does not implement reuse of objects and the responsibility definition are not well determined. [Awad et al. 2009] proposes a metamodel which focus on roles duties. The metamodel defines how to integrate OCL (Object Constraint Language) to process modeling aiming to determine participants responsibilities. It does not focus on solving ambiguities or other BPMN limitations. [Korherr and List 2006] extends BPMN and focus in measuring issues. Hence the BPMN was implemented without any constraints, which means that this metamodel has the same shortcomings of BPMN.

The proposed metamodel supports empirical processes modeling and execution, allows the users to change the modeled processes while they are running, promotes reusability of BP concepts defined using the metamodel, supports more than one participant to interact with the same task, relates four types of interactions with tasks (based on COBIT responsibilities definition), and implements recursive process modeling, execution of *ad hoc* processes and a wide variety of Process Patterns. According to BPMN, these features are challenges to the implementation of BP software suites [OMG 2009].

All these features show that the metamodel proposed and the respective execution engine enrich BP modeling for EIS domain and overcome limitations of BPMN related in literature. Our metamodel acts as a new brick to edify MDD approach for EIS with well-defined process support.

References

Achilleos, A., Georgalas, N., and Yang, K. (2007). An open source domain-specific tools framework to support model driven development of oss. In *Proc. of the 3rd European conference on Model driven architecture-foundations and applications*.

Almeida, A. C., Boff, G., and Oliveira, J. L. (2009). A framework for modeling, building and maintaining enterprise information systems software. *XXIII Brazilian Symposium on Software Engineering*.

Awad, A., Grosskopf, A., Meyer, A., and Weske, M. (2009). Enabling resource assignment constraints in bpmn. Technical report, Business Process Technology - Hasso Plattner Institute.

Bizagi (2010). Bizagi. Available in : http://www.bizagi.com/. Accessed: Jun 27 2010.

Bonita (2010). Bonita. Available in : http://www.bonitasoft.org/ . Accessed: Jun 29 2010.

Großkopf, A. (2007). An extended resource information layer for bpmn. *Hasso-Plattner-Institute for IT Systems Engineering*.

ITGI (2009). Cobit: Control objective management guidelines maturity model 4.1. /url-www.itgi.org. IT Governance Institute.

Korherr, B. and List, B. (2006). Extending the epc and the bpmn with business process goals and performance measures. In *Proc. of 9th International Conference on Enterprise Information Systems (ICEIS 07)*.

Mellor, S. J., Clark, A. N., and Futagami, T. (2003). Model-driven development. *IEEE Software Volume 20 - Number 5 - September/October*.

Nuffel, D. V., Mulder, H., and Kervel, S. V. (2009). Enhancing the formal foundations of bpmn by enterprise ontology. *Lecture Notes in Business Information Processing - 1865-1348 (Print) - Volume 34 - Advances in Enterprise Engineering III*.

OMG (2009). Bpmn specification 2.0. `www.bpmn.org`.

Russell, N., Hofstede, A. H. M. T., and Mulyar, N. (2006). Workflow controlflow patterns: A revised view. Technical report.

Smith, H. and Fingar, P. (2002). *Business Process Management: the third wave.* Megahan-Kiffer Press.

Systems, S. (2010). Enterprise architeture. Available: http://www.sparxsystems.com.au/. Accessed: Jul 01 2010.

Tsai, C.-H., Luo, H.-J., and Wang, F.-J. (2007). Constructing a bpm environment with bpmn*. *Proceedings of the 11th IEEE International Workshop*.

van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., and Barros, A. (2003). Workflow patterns. /urlwww.workflowpatterns.com.

Wesk, M. (2007). *Business Process Management: concepts, languages, architectures*. Springer.

zur Muehlen, M. and Recker, J. (2008). How much language is enough? theoretical and practical use of business process modeling notation. *CAiSE 2008. LNCS, vol. 5074*.

zur Muehlen, M., Recker, J., and Indulska, M. (2007). Sometimes less is more: Are process modeling languages overly complex? *2007 Eleventh International IEEE EDOC Conference Workshop (EDOCW'07)*.