FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Transformation of Business Process Models: A Case Study

Linda Anthuanett Norabuena Padilla

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: João Carlos Pascoal Faria (PhD)

January 20, 2014

Transformation of Business Process Models: A Case Study

Linda Anthuanett Norabuena Padilla

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

January 20, 2014

Abstract

Model-Driven Architecture (MDA) - the Object Management Group's (OMG) approach for Model-Driven Engineering (MDE) - is a visionary approach for software which has attracted the interest of a broad community of researchers and companies. In software engineering, MDA provides great advantages by reducing the effort required to implement systems, reuse existing assets and mainly provide a means to address the increasing complexity of software based systems. However, there are still challenges that affect the efficiency and applicability of the method. In a business environment, to apply MDA practices is a means to achieve quality and flexibility in the solutions that organizations provide to their customers.

The tendency to put into practice the methodologies of MDA has attracted the interest of organizations in the information technology industry. Sysnovare Solution Innovation SA is a concrete example of this trend. One of the competitive advantages of Sysnovare is a comprehensive Business Process Management (BPM) suite, developed in-house, that coves the full BPM lifecycle and employs some MDE techniques, namely the runtime interpretation of business process models. However, the BPM suite lacks a competitive visual modeling environment, not only for editing business process models but also for monitoring the execution status of process instances. Hence, the main goal of this dissertation work was to select, adapt and integrate an off-the-shelf visual modeling tool into Sysnovare BPM suite in a seamless way, taking advantage as much as possible of existing MDE/MDA technologies and standards, namely model-transformation technologies for assuring tool interoperability. Since the area of model-transformation is an active area of research and innovation, another equally important goal of this dissertation work is to assess the maturity and applicability of model-transformation technologies and standards, using the Sysnovare problem as a case study.

An adapted version of Draw.io application, built on the basis of the graphics engine mxGraph- a framework integrated in the BPM suite, will be the new graphical tool to model business processes. This choice allows implementing a graphical editor flexible and easily adaptable to integration's requirements. A bidirectional model-transformations between the representation used by the new visual modeling tool and the representation used by the BMP suite are performed in two levels. The first level is used to process input and output data developed with the transformation language Extensible Stylesheet Language for Transformation (XSLT) the goal of this layer is to convert XML files between the formats understood by the end tools (drawing tool and BMP suite) and the XMI format understood by the next level. Query/View/Transformation-Relational (QVT-R), on the contrary, is a language used in the second level to map the basic elements and modeling a business process. Medini QVT and Eclipse Modeling Framework are tools used to build the model transformation tool allowing to test the applicability of the approach.

After applying the model-based transformation as a means to assure the interoperability in BPM suite we obtain an efficient system integration. Through a set model transformation tools was possible to reduce the programming effort and to develop a transparent communication mechanism. Using the models transformation languages (QVT-R and XSLT) and their respective tools possible to verify the usefulness of MDA solutions.

Resumo

A arquitetura dirigida por modelos (MDA) - uma abordagem da OMG para a engenharia orientada a modelos (MDE) - é uma metodologia visionária no desenvolvimento de *software*, que tem atraído o interesse de uma ampla comunidade de investigadores e empresas. Na engenharia do *software*, o MDA provê grandes vantagens de forma a reduzir o esforço necessário na implementação de sistemas, reutilizar recursos e, principalmente, apresentar uma forma intuitiva para abordar a complexidade que a definição de um sistema requer. No entanto, ainda existem desafios que condicionam a eficiência e a aplicabilidade do método. Num ambiente empresarial, aplicar práticas MDA pode ser um meio para atingir qualidade e flexibilidade nas soluções que as organizações proporcionam aos seus clientes.

A tendência de levar à prática as metodologias do MDA tem vindo a captar o interesse das organizações na indústria da tecnologia da informação. A Sysnovare Innovation Solution SA é um exemplo concreto desta tendência. Uma das vantagens competitivas da Sysnovare é a *suite* de gestão de processos de negócio (BPM), que suporta o ciclo de vida completo de um processo de negócio e aplica algumas técnicas do MDE na interpretação dos dados para executar os modelos em tempo-real. No entanto, a BPM *suite* precisa de um ambiente de modelação visual competitivo, não só para a edição dos modelos de processos de negócios, mas também para a monitoração do estado da execução das instâncias de um processo. Desta forma, o objetivo principal do projeto é selecionar, adaptar e integrar uma ferramenta de modelação gráfica na Sysnovare BPM *suite* de uma forma eficiente, aproveitando o máximo possível das tecnologias e padrões do MDE/MDA existentes, ou seja, tecnologias para a transformação de modelos que assegurem a interoperabilidade da ferramenta. Finalmente, a transformação de modelos é uma área ativa de pesquisa e inovação, desta forma, outro objetivo igualmente importante do projeto é avaliar a maturidade e aplicabilidade das tecnologias de transformação de modelos usando o problema da Sysnovare como um caso de estudo.

Uma versão adaptada da ferramenta Draw.io, construída em base do motor gráfico mxGraph - uma *framework* integrada na BPM *suite*, é a nova ferramenta gráfica para modelar os processos de negócio. A seleção da ferramenta permite a implementação de um editor gráfico flexível e facilmente adaptável às necessidades da integração. A transformação de modelos bidireccional entre o modelo utilizado pela nova ferramenta de modelação gráfica e o modelo gerado pela BMP suite é realizada em dois níveis. O primeiro nível é desenvolvido com a linguagem de transformação XSLT e é usado para processar os dados de entrada e saída, o objetivo desta camada é converter os ficheiros XML entre os formatos compreendidos pelas ferramentas finais (editor gráfico e BMP suite) e o formato XMI utilizado pelo próximo nível. A linguagem utilizada no segundo nível para obter o mapeamento dos elementos básicos e a modelação dos processos de negócio é Consulta/Visualização/Transformação-Relacional (QVT-R). As ferramentas Medini QVT e Eclipse Modeling Framework são utilizadas para a construção da ferramenta de transformação de modelos, desta forma foi possível testar a aplicabilidade da abordagem.

Após aplicar a transformação baseada em modelos como via para garantir a interoperabilidade na BPM *suite*, obtemos um sistema de integração eficiente. Através de um conjunto de ferramentas de transformação de modelos foi possível reduzir o esforço de programação e desenvolver um mecanismo de comunicação transparente. Usando as linguagens de transformação de modelos (QVT-R e XSLT) e as suas respectivas ferramentas foi possível verificar a utilidade das soluções do MDA.

Acknowledgements

I would like to thank the contribution of my supervisor, Professor João Pascoal Faria, from the Faculty of Engineering of University of Porto, for the advice and guidance throughout the project. Also, my recognition to engineer Antonio Cunha and to Sysnovare Innovation Solution SA team for presenting the challenge and helping to find the focus of the investigation.

I am also grateful to Sysnovare for the provision of the firm facilities and the availability to enable the development of the project in an enterprise environment. Finally, I thank all the people who offered advice or helped in any way throughout the project phase.

Linda A. Norabuena Padilla

"The best way to get a project done faster
is to start sooner."

Jim Highsmith

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

AAF	Assistant of automatic forms
ATL	ATLAS Transformation Language
BP	Business Process
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMI	Business Process Management Initiative
BPMS	Business Process Management Suite
BPMO	Business Process Modeling
BPMN	Business Process Model and Notation
CWM	Common Warehouse Metamodel
EMF	Eclipse Modeling Framework
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
GG	Graph Grammars
GUI	Graphical User Interface
MDE	Model-Driven Engineering
MDA	Model-Driven Architecture
MDD	Model-Driven Development
MOF	Meta Object Facility
MVC	Model View Controller
OMG	Object Management Group
OS	Operating system
PEE	Process Execution Engine
PIM	Platform-Independent Model
PL/SQL	Procedural Language/Structured Query Language
PSM	Platform-Specific Model
QVT	Query/View/Transformation
QVT-R	Query/View/Transformation Relational
RPM	Repository of Process Model
TGG	Triple Graph Grammars
UI	User Interface
UML	Unified Modeling Language
WF	Workflow
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language for Transformation

Chapter 1

Introduction

1.1 Context

The usage of models as first class citizens in the development of complex systems has increasingly been applied and studied in software engineering. A practical example of the application of Model Driven Development (MDD) is the paradigm of Business Process Management (BPM). In BPM a model is a representation of a business process within an organization; this model can be implemented in an information system, allowing the interaction between applications and users so that they can perform all the activities that a particular business process requires.

Sysnovare Innovation Solutions SA, referred to as Sysnovare throughout the document, has been using the best practices of MDD in its products. An example of it is the Sysnovare Business Process Management Suite, a web platform that can manage processes of some business organizations in Portugal and abroad. However, there has been a growing interest to improve the competitive advantages of Sysnovare solutions, therefore emerging the need to investigate the application of more methods of MDD in products that support Business Process Management.

Integrating MDD and BPM approaches in the context of Sysnovare will provide even more comprehensive solutions to customer needs. The combination of these paradigms, nowadays increasingly investigated by the academic and business society, will allow coming to a proposal for the project's main goal: to improve the modeling of business processes in Sysnovare BPM suite, replacing the graphical modeling tool by an existing application that specialized in this area. In order to arrive at a solution for the project the dissertation work encompasses analyzing and selecting existing modeling tools on the market and creating an automated communication system between the BPM suite and the modeling tool based on MDD techniques and technologies.

Thus, Sysnovare offers one of its "new generation" products as a case study for the development of the project. The BPM suite, a product widely used in everyday life of organizations with activities in the areas of human resources, enterprise resource planning and academic management, will be the basis artifact of research. The suite is a starting point for defining requirements and limitations of the project.

1.2 Problem

In systems engineering, modeling a business process is the activity of representing a process of a company. Through abstraction of the behavior of a business it is possible to analyze and improve the way how the process will perform its various activities. In a system that supports Business Process Management (BPM), modeling is the first phase which a business process must go through; the efficiency of this process will depend on this first phase, influencing the later stages, which will allow executing the process in the same system.

Sysnovare BPM Suite is an innovative product that provides an agile solution in the process management of an organization, also ensuring a complete monitoring by combining other resources and internal or external tools to the organization. Currently, Sysnovare aims to improve the modeling component of its BPM suite. The purpose is to provide a better modeling experience suite to final users, such as consultants, analysts and managers. This way, one can speed up the modeling step in the definiton of a business process, increasing the usability of the current graphical modeling tool and avoiding that users resort to other graphical tools.

Thus, Sysnovare launched the challenge of finding a viable and fast solution to be adapted in a web environment so that the suite communication mechanism with other external systems is improved, in this case with a modeling tool that allows to be integrated in Sysnovare solutions and may permit the replacement of the current modeling tool. This process requires an integration of systems that enables transparent, efficient and automatic communication, since in a web environment users require fast and consistent answers. Moreover, to achieve the main objective of the project it is essential to introduce model transformation, as each system has models that obey different data structures and need to be transformed so that each system can interpret data from its own perspective.

Interoperability between systems through the transformation of models includes the exploration of techniques and recent technologies in this area so that automated systems result as response, systems that reduce the implementation effort and the amount of errors that can exist in an implementation with a lower level of abstraction. Besides, before defining the system integration it will also be necessary to identify the modeling tool/framework that will improve the user experience and that at the same time is highly flexible so that the application can be customized to the needs of Sysnovare. The final result of the integration allows users to enjoy a more complete answer in the management of business processes, thereby increasing the competitive advantages of BPM Sysnovare suite in the BPM market.

1.3 Objectives and Contributions

The main objective of the project is to integrate a modeling tool for business processes that allows replacing the current modeling component of Sysnovare BPM suite. This way, it is intended to improve the BPM suite in efficiency, usability and at the same time gradually evolve from a proprietary notation to a standard process modeling notation. In the modeling phase, the new tool allows the user to easily create process models and define a basic set of attributes of each graphic element, which can be complemented at a stage prior to the implementation process. To achieve the main objective it was necessary to accomplish two sub-goals that ultimately will lead to the desired solution.

Thereby, the first objective is to analyze a set of graphical modeling tools that allow modeling business processes. For this purpose it was necessary to analyze the distance between Sysnovare proprietary notation and BPM standard notation, and it is essential to define a level of conformity between the notations and select a group of graphic elements that will be supported in the communication systems. Later, a modeling tool will be integrated into Sysnovare BPM suite.

To establish a transparent communication between the two business tools and integrate their functionality, there was the need to analyze the existing approaches and technologies that are applicable in practice and allow them to be adapted to the requirements of Sysnovare. Thus, the second objective is to define a mechanism for communication among tools while maintaining the best practices of Model Driven Development.

As BPM suite is developed in a web environment, it will require a definition of a fast approach in terms of responses to the user. It is the user who, after modeling a business process in an integrated graphical tool, will have as expectations the pursuit of management phases in simple and quick steps, and communication between systems will be noticeable through the graphical interface.

It will also be necessary to work in order to achieve dynamism and timely response involving the requirements of web applications as mentioned. Responses in real time, currently present in the approach of Sysnovare suite, will be requirements in the definition of the solution. Thus, in the research of technologies we will seek a solution that is adaptable, easy to integrate, and that allows evolving in the communication with other tools or achieving future goals for the Sysnovare products.

Defining the technologies needed to establish communication between two systems and a well-defined approach for integrating the modeling tool in BPM suite, it will be possible to present a case study for the research community in this area, i.e. transform models of business processes in a business environment. This way, interoperability will be introduced to Sysnovare with external systems, contributing to the modeling performance of a business process, streamlining the design process for consultants and analysts, avoiding the use of external tools and improving part of the monitoring of the execution status of process instances.

During in the dissertation work was produced other technical report. The content of this report has been removed because the document is limited in amount of pages. Thus, the information excluded from this document is a more complete study of the standard notation of BPM and a detailed analysis of the modeling tools.

1.4 Document Structure

This document is organized into 8 chapters, as follows.

The first chapter brings forward the context of the problem presented by Sysnovare, the objectives to be achieved and the motivation that guided the development of the project.

The second chapter presents the analysis of the problem, giving an overview of Sysnovare BPM suite, the product competitive advantages and the limitations of the modeling component of the suite. This chapter will allow understanding the bases of the decisions which may lead to the solution of the problem. At the end of the chapter we detail the challenges that have motivated this project.

The background and the review of the existing literature on the model transformation of business processes are described in the third chapter. The chapter contains all the necessary information related to the two main themes of the project: Model-Driven Development and Business Process Management.

In chapter four we present an overview of the architecture of the proposed solution. Before reaching a solution, in the same chapter we present detailed analysis of all the data to be communicated and the selection of techniques and technologies that are used in the following chapters to implement the solution.

Chapter five describes the development of the Visual Process Modeling Environment. We describe the implementation carried out to integrate mxGraph framework in BPM suite and the specification of the creation of the graphical editor based on the modeling tool Draw.io.

The work to create a tool for model transformation is described in chapter six. Similarly, we describe the components that were added to the modeling component of BPM suite.

After presenting the mechanism of interoperability, in the seventh chapter we describe two examples of modeling in order to evaluate the usability and performance of the proposed approach.

Finally, the eighth chapter presents an overview of the work undertaken and described throughout the document, the results obtained in the project and the work needed to improve the modeling of a business process in Sysnovare.

Chapter 2

Problem Analysis

In this second chapter we focus on the Sysnovare BPMS: main features and communication mechanisms that allow information interchange among the modeling component and other systems that compose the suite. Furthermore, we introduce the modeling process; so it is important to know the approach of Sysnovare solutions.

In the research to improve the modeling component of Sysnovare BPMS there emerges the need for analyzing this component. Thus, we identify its limitations and requirements so that a solution may be found to replace the existing modeling component.

Finally, this chapter concludes with the challenges that the project requires to define a solution based in Model-Driven Engineering (MDE) techniques. MDE will allow maintaining the competitive advantages that Sysnovare solution currently owns.

2.1 Sysnovare BPM Suite

Sysnovare BPM suite is a solution to support Business Process Management (BPM) allowing an agile process modeling, automatic process execution, process execution monitoring and improving the process in the optimization phase (see Figure 2.1). This tool is an independent web platform designed to ensure an adaptable system where each organization can specify their business processes. The simplicity of this solution allows the user to improve the process models without resorting to expert advice. Therefore, the suite acts autonomously and can be integrated with other applications of the organization.

Sysnovare suite is composed by 5 independent products: Workflow used to support all phases, Information Structures to define roles in modeling's phase, Assistant of Automation Forms to associate user interfaces with activities, Document Management and Centralized Entities for document management created in the execution's phase. The components allowing the organization to choose the systems that can improve performance and strategies for controlling their processes.

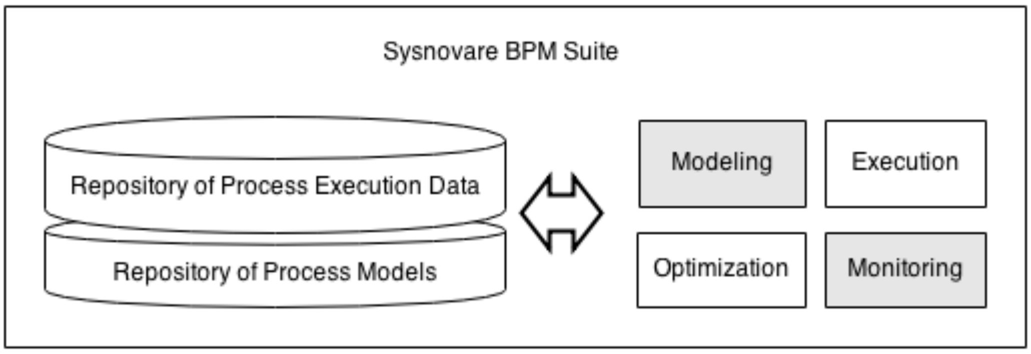

Sysnovare BPM Suite
Repository of Process Execution Data
Repository of Process Models
Modeling
Execution
Optimization
Monitoring

Figure 2.1: BPM phases supported by Sysnovare BPM suite

Workflow's component facilitates the exploration of information by the user in a simple and contextual way, regardless of the stage of the process. When the stage of the process involves the collection of data and / or preparation of documents, this solution automates the construction of interfaces to interact with the user. Through each component, a process's phase interacts with the repositories.

Some of the main features of the suite are the following:

- Process modeling and team modeling - Workflow
- Processes monitoring - Workflow
- Repository of the documentary information - Document Management
- Search information - Document Management
- Elaboration of documents - Assistant Automatic Forms
- Building information collection forms - Assistant Automatic Forms
- Parameterization of structures - Information Structures
- Registration of entities - Centralized Entities

Sysnovare suite has as strengths executable models in Process Execution Engine (PEE) sub-component and automatic construction of forms in Assistant of Automation Forms (AAF) component (see Figure 2.2). Workflow encompasses Model-Driven Development (MDD) practices, depicting a system that is robust, adaptable and easy to maintain. PEE component uses an interpretative MDD technique allowing the execution of process models in real-time.

However, as weaker points, the Process Modeling Tool sub-component is limited in terms of usability and it follows a proprietary notation, which is misaligned with the standard Business Process Modeling and Notation (BPMN). Furthermore, a modeling tool builds a graphical representation because it works directly with a repository of process models, not storing a graphical model representation. This last limitation influences the monitoring phase, so that there is no

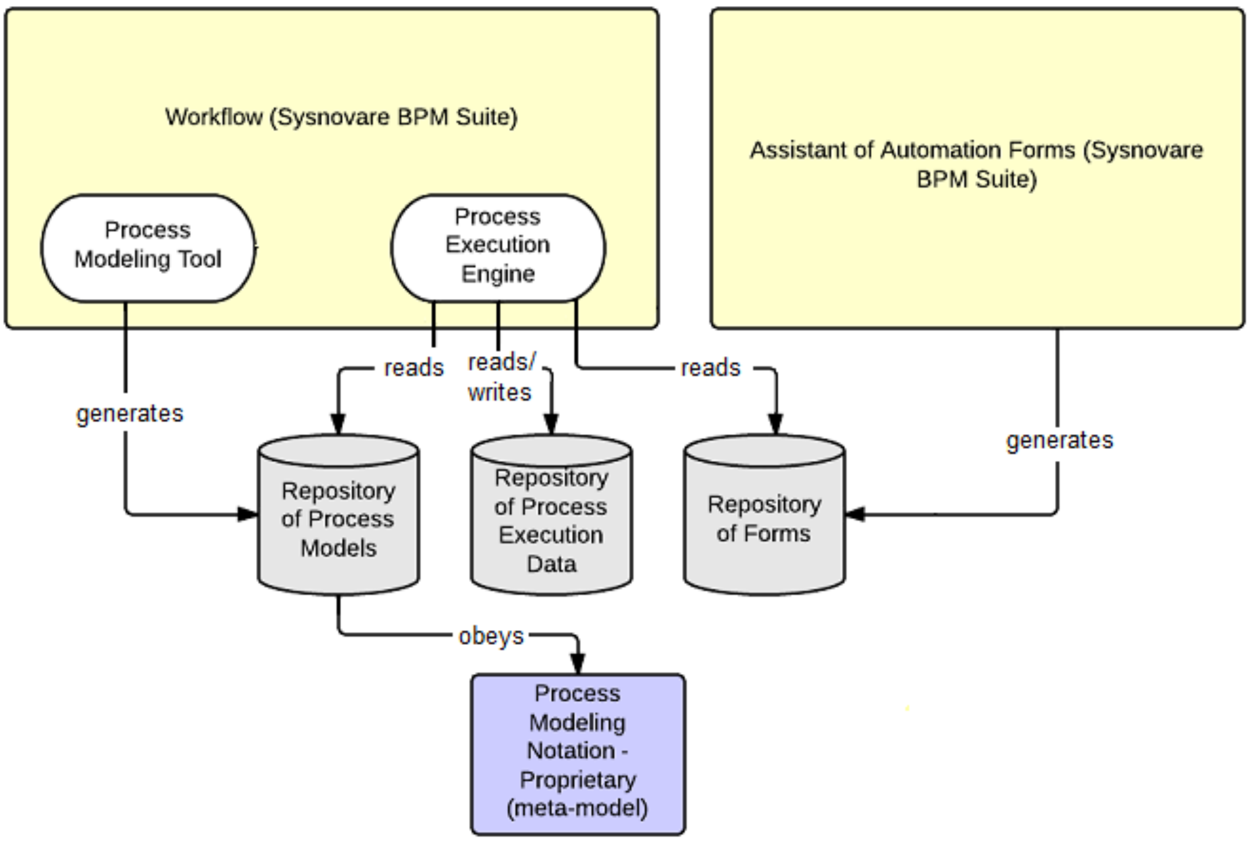

Workflow (Sysnovare BPM Suite)
Process Modeling Tool
Process Execution Engine
Assistant of Automation Forms (Sysnovare BPM Suite)
generates
reads
reads/writes
reads
generates
Repository of Process Models
Repository of Process Execution Data
Repository of Forms
obeys
Process Modeling Notation - Proprietary (meta-model)

Figure 2.2: Partial Architecture of BPM Suite

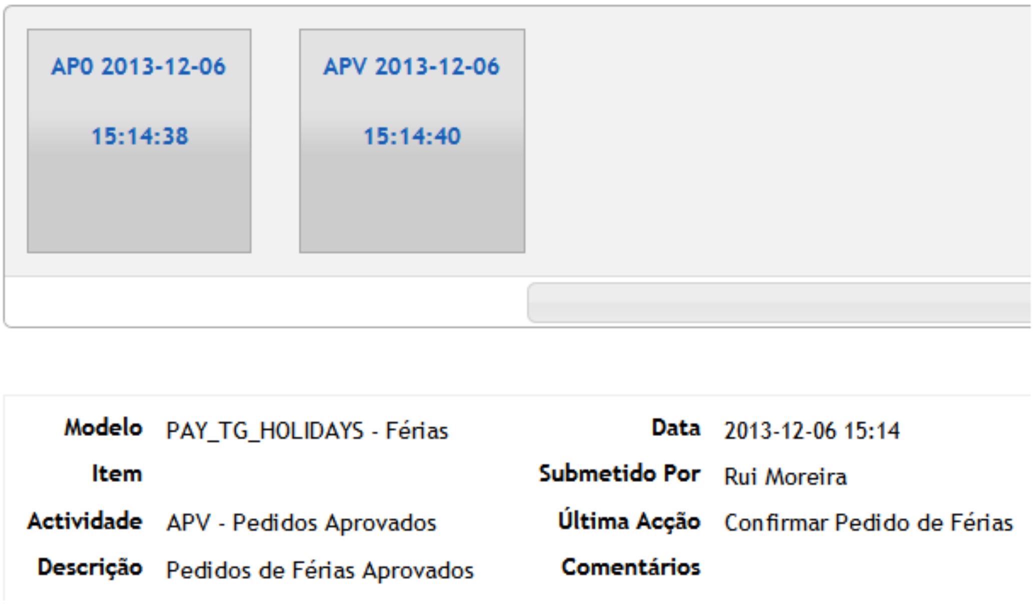
graphical view to control the model. Currently, monitoring is presented in text and graphically in a historic format as we can see in Figure 2.3.

As we mentioned in the beginning of this section, Sysnovare suite is a platform for information exchange between each phase of the business process management lifecycle: modeling, execution, monitoring and optimization. The communication is established via a shared repository as shown in Figure 2.2. Sysnovare BPM suite is a PL/SQL web application that basically consists of a set of procedures that interact with the database, browsers and backend, a set divided in three logical parts according to the Model-View-Control (MVC) architectural pattern: the model, views and controller parts.

Finally, throughout this project the suite component that will be used is: Workflow (WF). The WF component will be complemented with a new solution for business process modeling that will possibly influence in the visualization of the monitoring phase.

2.2 Process Modeling

Modeling a process is the first step in managing business processes; it is the main phase that will allow defining an abstract representation of a business process. If the model is not well analyzed at the modeling phase, errors can occur in the next phases; therefore, it is important to use intuitive



Modelo	PAY_TG_HOLIDAYS - Férias	**Data**	2013-12-06 15:14
Item		**Submetido Por**	Rui Moreira
Actividade	APV - Pedidos Aprovados	**Última Acção**	Confirmar Pedido de Férias
Descrição	Pedidos de Férias Aprovados	**Comentários**	

Figure 2.3: Sysnovare Business Process Monitoring User Interface (in portuguese)

tools with a high level of usability so that the user can easily understand the logic and behavior of a business process.

Process modeling basically describes in a graphical notation how the business should behave, or in other words, how the process will fulfill one's goals and tasks. A simple process can be broadly engaging, i.e. it may need to support recursiveness in some sequential or parallel tasks, to define decisions based or not on conditions for the workflow to continue its route or to need the participation of several actors, such as people, organizations or applications to execute sub-tasks.

The modeling component of the BPM suite enables modeling business processes with two types of graphical objects. In Figure 2.4 we can see that the boxes represent an activity that may be associated with other elements through a arrow; activities can also change color depending on type. As we can see, the graphical tool is limited and unintuitive in the graphical representation, the design does not allow viewing any graphical component that distinguishes the behavior characteristics of the elements; an example would be an icon of a person in the activity *"Preenchimento de requerimento"* (Filling in application). This way, it would be clear that the activity needs some user intervention. In addition, there are implementation errors in the graphical tool, changing the visualization of the diagram.

The modeling tool draws the diagram at the interface querying the model elements in the model repository: activities and transitions. After receiving the information, the application draws the objects and changes the colors of activities, depending on their type.

Currently there are several tools for modeling business processes; however, there is a very wide variety of tools that may or may not obey standard notation: Business Process Modeling and Notation (BPMN). The tools can obey a proprietary notation to cover all the requirements of a specification, or simply obey a specific notation version. Given that modeling is an isolated

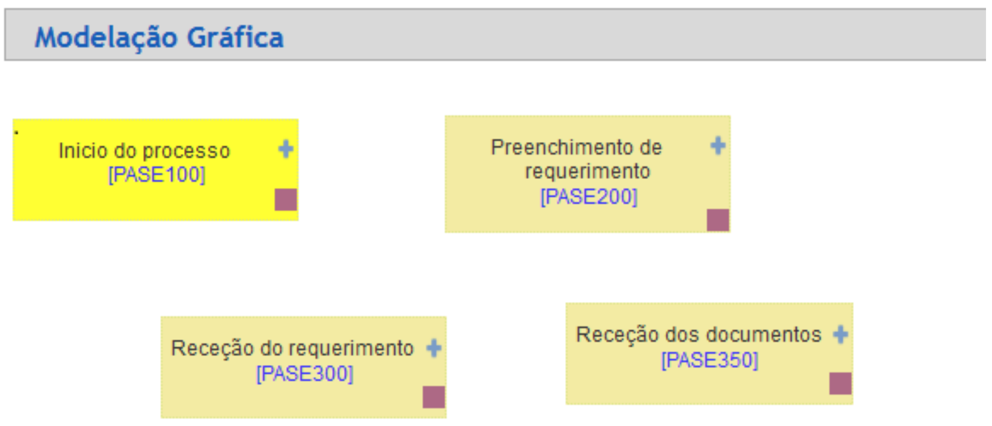

Modelação Gráfica
Inicio do processo [PASE100]
Preenchimento de requerimento [PASE200]
Receção do requerimento [PASE300]
Receção dos documentos [PASE350]

Figure 2.4: Sysnovare Business Process Modeling User Interface (in portuguese)

phase from the other phases of a BPM process, most of the tools exchange information through extensible markup language (XML). The XML files can be useful for importing/exporting models in the BPM suite.

2.3 The Challenge

After the overview of how Sysnovare BPM suite works and supports process modeling, in this section we analyze what can be improved in the modeling component.

Replacing the modeling component of BPM suite by an external modeling tool can be a demanding task, that requires the careful identification of requirements and selection criteria, and the rigorous assessment of risks associated with candidate solutions. Finding the balance between the different criteria and risks associated to selection is part of the challenge of Sysnovare.

Selecting a graphical tool must meet the requirements of the case study, the application must adapt to the needs of integration in BPM suite, such as: engine import / export of models via XML files, customizing the graphics, so that the visualization of workflow presents the graphical characteristics of the behavior of elements; extra actions to draw diagrams in order that the modeling experience is more dynamic; maintain efficiency in the creation of graphic elements and improve the characteristics of the elements using the forms and styles of BPMN notation.

The way to establish communication between two systems that generate models according to different metamodels will be another challenge. First we will need to choose the information to be interchanged, i.e. to define a basic set of elements of the notation. As many modeling tools obey the standard BPMN notation, it will be necessary to analyze the distance of the proprietary notation of Sysnovare with respect to BPMN. Defining compliance levels may be useful to reach a final conclusion and select a group of basic elements that permit to test the applicability of the approach.

Finally, both systems allow the information exchange of models via XML files, so an XML interpreted in BPM suite must be transformed so that the modeling tool can represent the diagram. The XML data follow a data structure previously defined by the tool, so the transformation from one model to another will lead again to finding a balance between the different criteria and the risk of not choosing the best selection. Evaluating the different techniques and technologies for model transformation will be a process that will require more effort, as integration should involve a combination of several mechanisms that allow: efficiency, data consistency and data sharing in both directions so that the models created in the suite can be processed by the system and displayed in the graphical interface. Moreover, in this process maturity of the approaches will be a key part in the final decision of a proposal and to define the architecture of the approach.

Being aware of how big a challenge in this area can be, since there are many perspectives to consider, Sysnovare knows that interoperability in BPM suite requires a thorough discussion and analysis.

2.4 Conclusion

To find a feasible solution it is important to understand the problem and identify the context in which it appears. After knowing the Sysnovare suite better, we can say that the BPM suite is a flexible and agile solution to manage business processes. Since the suite is a web platform, it should provide consistent and quick answers to users.

The BPM suite uses as development methodology the most recent paradigms related to Model-Driven Development (MDD), to enrich the competitive advantages of the product and allow generating responses to users in real time. However, currently the suite needs to be complemented in process design component to improve its competitive advantage in a BPM market that is constantly growing. After recent studies related to the standard modeling notation BPMN have arisen [ARC+13], the Sysnovare began to take interest in knowing the notation and discussing the possibility of establishing an integration mechanism between the suite and other modeling tools.

To improve the modeling component, this component should be replaced by external graphical tool. Given that collaboration among applications is used nowadays, it is necessary to establish a rapid and efficient mechanism to set a clear communication among BPM suite and new graphical tool with different data structures or notations. Integration or collaboration among applications must be an automatic mechanism so that it is efficient and there are no errors.

Apply MDD methods has been evaluated and approved by the software engineering community. The good practices of MDD are often related to the strategies of enterprises regarding costs, time for research and trends. In this way, it will be possible to analyze MDD methods in the BPM context, presenting a real study case that allows automating the communication among tools.

Chapter 3

Background and State of the Art

This chapter gives an introduction and a review of relevant literature helpful to design a solution for the problem at hand. Therefore, throughout this third chapter, we focus in two main issues: Business Process Management and Model-Driven Development.

Recent research works in the field of software engineering involve model-based transformation approach to interchange information among phases of the business process management lifecycle. Also, transformation among models can be useful for interoperating systems, where the information created or obtained in one tool can be used in other systems.

3.1 Business Process Managament

Aalst et al. [AHW03] define BPM as follows: "Supporting business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information". Therefore, in other words, we can define BPM as an approach which enhances an IT organization ability for improving business performance outcomes.

3.1.1 BPM Lifecycle

Generally, business process management activities are grouped in five categories: design, modeling, execution, monitoring and optimization. In Figure 3.1 we can see a standard BPM lifecycle, where each cycle phase can be repeated in both senses of the sequence design-optimization.

Current BPM systems mainly support modeling, execution and monitoring phase; these phases and others are described as follows:

- **Design** process that identifies the existing processes and the processes that must be designed. This phase aims to ensure a correct theoretical design for building an efficient process and thus reduce the errors over the process lifetime.

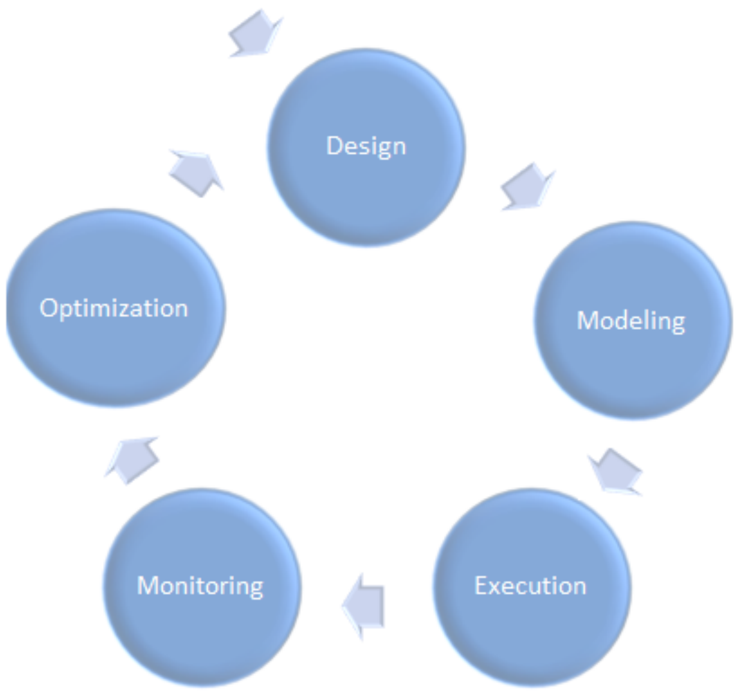


Figure 3.1: Business Process Management Lifecycle

- **Modeling** carries a theoretical design to a process abstract representation involving tools, business analysts, executive managers and others interveners for process improvement and quality management.
- **Execution** is a phase in which we can automate resources and activities of a business process. This information is required for the execution of a model according to the formally defined model in the preceding phases. Generally, a process execution is a combination of software and human intervention.
- **Monitoring** allows controlling individual process instances. This phase is useful to know what activity is being executed during workflow, statistical data that can be obtained, timings and process behaviors.
- **Optimization** includes enhancements in the process design after an analysis of process behavior in the phases of modeling and monitoring. In this phase, process performance information is obtained.

According to the strategic vision and needs of an organization, the standard lifecycle is adapted by the enterprise.

3.1.2 BPM Suites

According to Gartner's research [Gar13], the following trend is observed during 2008 - 2010 [SH10]: "more organizations are adopting BPM as a discipline and scaling up their efforts to establish BPM as an enterprise program". As a consequence, BPM market is becoming a competitive one with different proposals for buyers and the IT needs of organizations.

A BPM suite is a technology platform that supports all phases of business process management: modeling, execution, monitoring and others, as it was mentioned in the previous section.

Gartner's 2010 Magic Quadrant [SH10] evaluates the top 25 vendors in the BPM market and presents four scenarios for categorizing each BPM suite: challengers, leaders, niche players and visionaries. Each category allows obtaining a general vision of a BPM suite: leaders - a suite has an ability to support iterative process improvement; visionaries - a suite proposes innovative ideas; challengers - it is necessary to make major changes for organizational structures alignment; niche players - the company product is new for this market.

Figure 3.2 shows suites that will be presented in the following listing of leaders in BPM solutions: Pegasystems, IBM, Progress, Appian and Software AG.

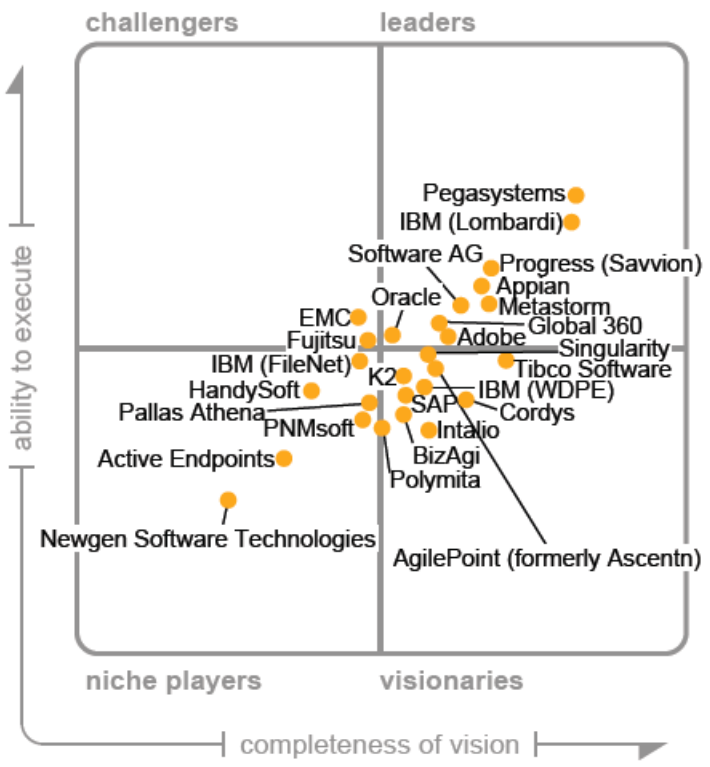


Figure 3.2: Magic Quadrant for Business Process Management Suites [SH10]

More and more BPMSs are integrated in the organizations because they provide a set of technologies that give a consistent user experience throughout the business process lifecycle. BPM suites continue to improve and grow because companies are investing more to be leaders on business process management.

3.1.3 Business Process Modeling and Notation

Business Process Modeling (BPMO) is frequently used in software engineering [BRU00]. BPMO is an activity which represents a business process of an organization; so in recent years organizations tend to adopt new tools that allow the implementation and control of their strategies in process management improving their performance. In this project we focus on the modeling phase of a process lifecycle that may obey a standard notation or not.

A notation is required to create visual process models, including different elements that represent the order/sequence of the activities a business process should follow [All10]. Thus, the standardization of notation to represent a process allows using a familiar language for all users and collaborators that work in the BPM domain as consultants, designers, analysts and customers.

Business Process Model and Notation (BPMN) defines a formal standard notation that was developed by the Business Process Management Initiative (BPMI) and is currently maintained by the Object Management Group (OMG). The latest version of this standard notation is BPMN 2.0 [OMG11]; this specification provides an extensible notation in order to allow customization of elements according to specific needs of organizations. For that purpose the specification allows adding attributes to each element and extending elements. In the Figure 3.3 was created with Eclipse Modeling Framework Project based on the official notation, BPMN2. In this figure we can see all the elements of BPMN2 and relations among the elements.

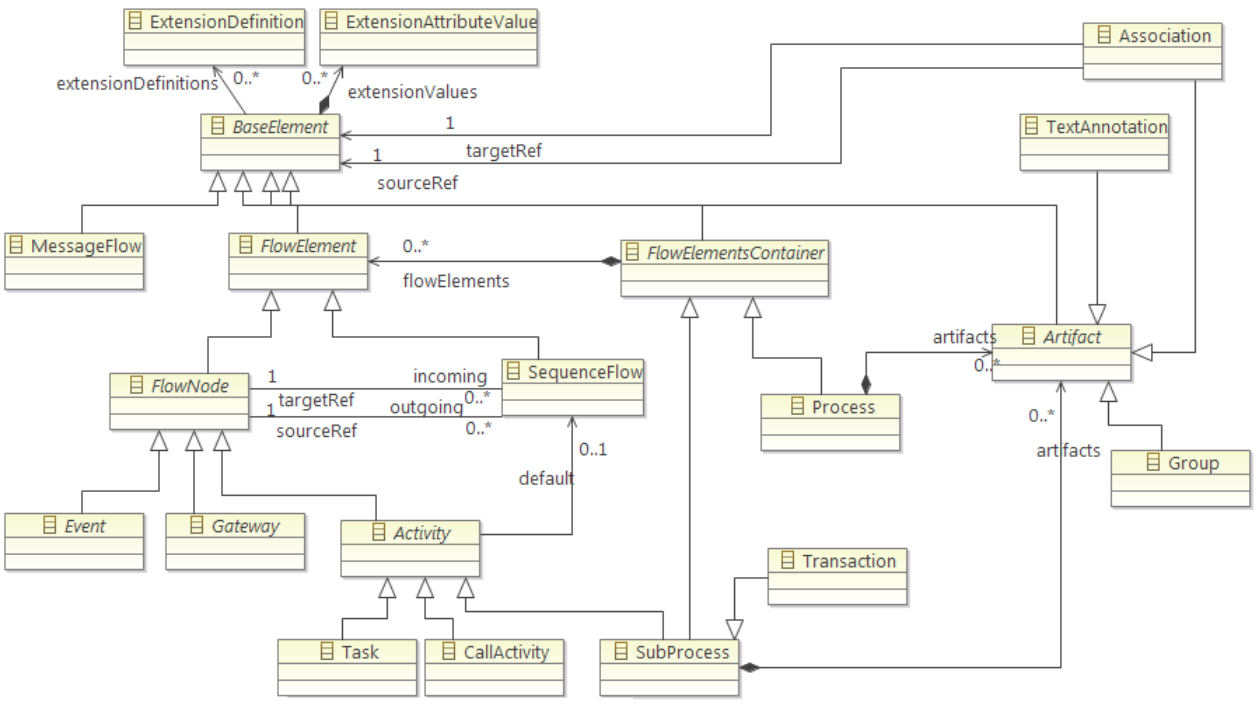


Figure 3.3: Partial view of the BPMN2 metamodel (Ecore notation)

Some BPMN elements are described as follows for understanding the basis of the language:

- **BaseElement** is the main element in the BPMN metamodel; the other elements inherit the properties of this object. The additional attributes can be defined with an *ExtensionAttributeValue*. To define a new object in the notation, the *BaseElement* can be associated with an *ExtensionDefinition*.

- **FlowElementCointainer** is an element that contains flow elements and it can be a *Process* or *Sub process*.
- **FlowElement** is an element that describes the behaviour of a process. It can be *FlowNode* or *SequenceFlow*.
- **FlowNode** is an activity, event or gateway included in a process. The node helps defining the behaviour of the business process.
- **SequenceFlow** is the association between a source and target *FlowNode*. It defines the ordering or sequence of flow.
- **MessageFlow** is the association between two participants in collaboration.
- **Artifacts** are objects that provide a visual mechanism, adding information such as annotations, group of objects and associations for data objects (input, output, collection and store).

Figure 3.4 shows part of a Sysnovare Business Process, redesigned in BPMN2 notation using the Bizagi Modeling Tool. In Section 7.1 we can see a description of vacation request process that is a common and internal process of many organizations.

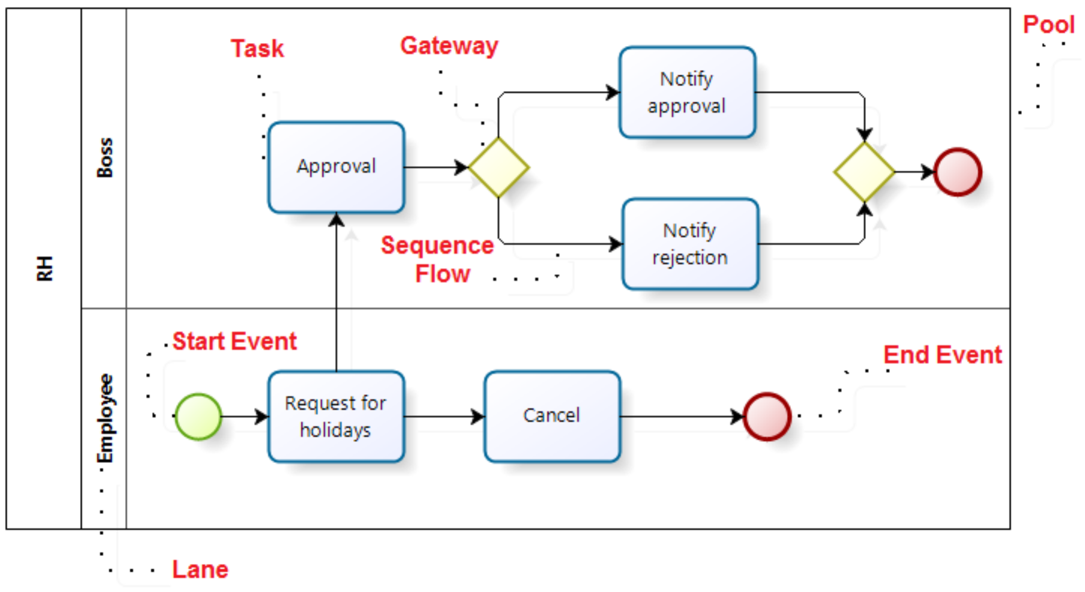


Figure 3.4: Example of Business Process: Vacation Request (created with Bizagi Process Modeler)

3.1.4 Process Modeling Tools

Today, there are more than 40 modeling tools in the BPM market. Software has been used to simplify the modeling phase, ensuring mainly the following features: an easy process design; graphical symbology standards - such as notation BPMN; additional features to complement activity information (input/output data, comments, groups, etc.); mechanisms for publishing information in collaborative environments and tools for integrating with other systems, such as databases.

Some modeling tools adopt the standard notation BPMN 2.0 or a previous version. However, many tools do not show compliance with all the aspects of the specification. In fact, current modeling tools provide heterogeneous semantic and different conformance levels to BPMN, introducing limitations in the communication among BPM systems [Paw11].

Finally, according to the needs of the organization and business strategies, the companies choose a modeling tool to embed into their systems or simply to establish interoperability with other systems in order to reach a greater number of customers. Some modeling tools are: ADONIS, Bizagi Process Modeler, IBM WebSphere Business Modeler 7.0, TIBCO Business Studio, Draw.io, Intalio, No Magic, MID Innovator for Business Analysts, Oracle Business Process Management Suite 11g, JBPM5, BPMN Web Modeler, Bonita Soft, Activiti BPM Platform, Lucidchart, Signavio Process Editor and others.

3.2 Model-Driven Development and Model Transformation

In this section an overview of Model-Driven Development will be presented, and specifically Model-Driven Architecture (MDA), which is a standard that aims to isolate the business logic of the application and facilitate maintenance and evolution of technology. The MDA framework models are transformed through a tool; the same tool or another then turns the final model into the source code.

A new research area for new specific tools for model transformation or code generators for model interchange is being studied. Thus, in this section, such techniques and technologies will be presented so that in the next chapter they are analyzed on the basis of a set of criteria.

3.2.1 MDD Approaches

Model-Driven Development (MDD) supports Model-Driven Engineering (MDE) and both are generic terms which describe the systems as models that conform to metamodels and may be manipulated through model transformations [Lid11].

MDD is a methodology to implement computer programs quickly, efficiently and with minimal cost. MDD approach enables employees to work together on a project, even if their individual experience levels varies greatly. It allows an organization to maximize the effective work on a project, minimizing the overhead required to produce software. There are two main methods in MDD: code generation and model interpretation [TVZ12]:

- Code generation is generally used to describe the problem's domain and give as a result a program code which must possibly be adapted manually. In this method the modeling environment is completely separated from the execution environment.
- Model interpretation has the ability for changes to be effective and visible in real-time; it is the main advantage of interpretive MDD. The systems can be adapted instantly because the execution engine interprets a model one part at a time.

MDD is based on models because it allows simplified representations of a particular system.

3.2.2 Model-Driven Architecture

MDA launched by OMG can be considered as an instance of MDD based on the core standards of OMG such as Unified Modeling Language (UML), MetaObject Facility (MOF), XML Metadata Interchange (XMI), and the Common Warehouse Metamodel (CWM). Basically, MDA focuses on architecture and on models to build a system which allows investigating the usefulness of models as a contribution to increase abstraction and to provide different perspectives.

The central point of Model-Driven Architecture is the use of models in the software development process. In this context models should describe the representation of systems consistently, accurately, and with all the sufficient information. In MDA approach, models can be at different levels of abstraction and described in different languages. In Figure 3.5, we can see that there are usually 4 layer models.

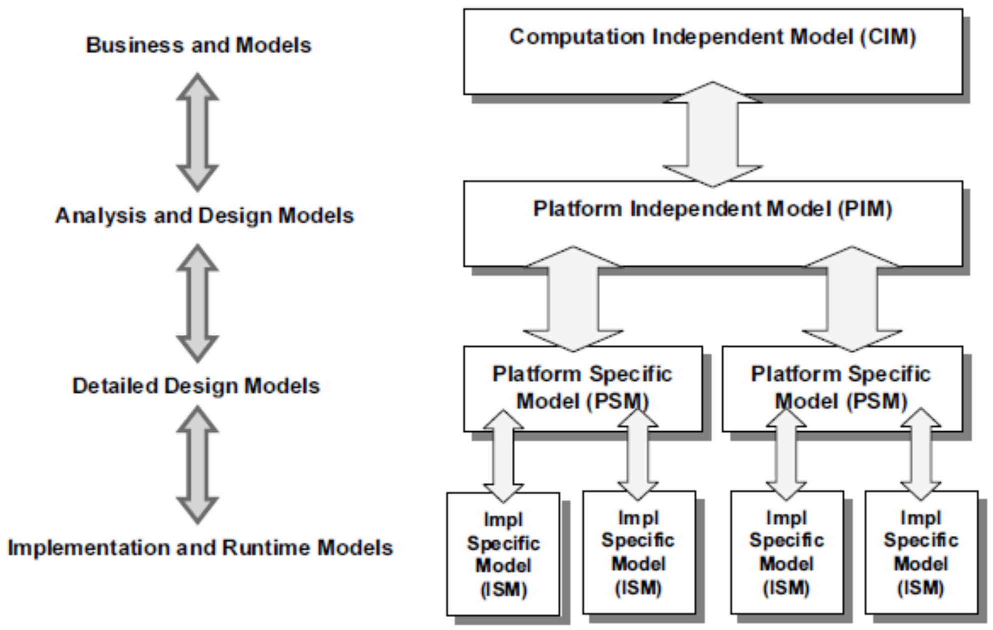

Business and Models
Analysis and Design Models
Detailed Design Models
Implementation and Runtime Models
Computation Independent Model (CIM)
Platform Independent Model (PIM)
Platform Specific Model (PSM)
Platform Specific Model (PSM)
Impl Specific Model (ISM)
Impl Specific Model (ISM)
Impl Specific Model (ISM)
Impl Specific Model (ISM)

Figure 3.5: The layers and transformations of MDA [BCT05]

In summary each layer is as follows:

- Computation Independent Model (CIM) is often known as a business or domain model. It presents exactly what the system should do; however, it does not specify the related information technologies.
- Platform Independent Model (PIM) displays a model of a software system independent of the technological platform that will be used to implement it.
- The next level Platform Specific Model (PSM), by contrast, combines the specifications of the PIM with the details needed to determine a model that is linked to a specific technology platform.
- Finally, the code is generated through a transformation that allows implementing the system.

In a model-based system, to move from one level to another, models must be transformed; this way, the tools offer the user the flexibility to direct the steps of the processing to his specific needs.

Modeling is a way of programming in software engineering. Recent studies are focused in notations and technologies that allow users to express system perspectives that can be implemented in the development phase according to a model designed or directly mapped into a specific programming language code.

Unified Modeling Language (UML) and Business Process Modeling and Notation (BPMN) are modeling languages that enable the construction of a model capturing the important features of a system. In this project, BPMN can be considered as the model source (PIM) to be transformed to the target model (PSM), in this case the Sysnovare workflow model.

3.2.3 Model Transformation Techniques and Technologies

In this section an overview of model transformation is described. In the last years several approaches have been explored, technologies and languages which allow transforming a source model into a target model. Murzek and Kramler [MK07] define that the application of transformation on Business Process Modeling (BPMO) is a challenge because the current languages provide general solutions that do not cover specific issues in distinct areas of horizontal business process models. Therefore, it is necessary to analyze the languages that can support the business process of the case study.

This paradigm involves modeling, metamodels and mainly model transformations [Ken02]. Then, the basic concepts of a model transformation are described as follows:

- Source/target model.
- Metamodel of source and target: it defines the structure of models.
- Transformation rules: it is an optional component that depends on the applied technique.
- Transformation engine: it reads a source model, executes a transformation and writes the output model.

Model transformation involves two levels of abstraction (see Figure 3.6), a higher level of abstraction where are defined the model structures (metamodels) and transformation rules that describe the mapping between models, and a lower level where is instantiated a source, target model and a transformation engine that executes the rules to transform the models [Str08].

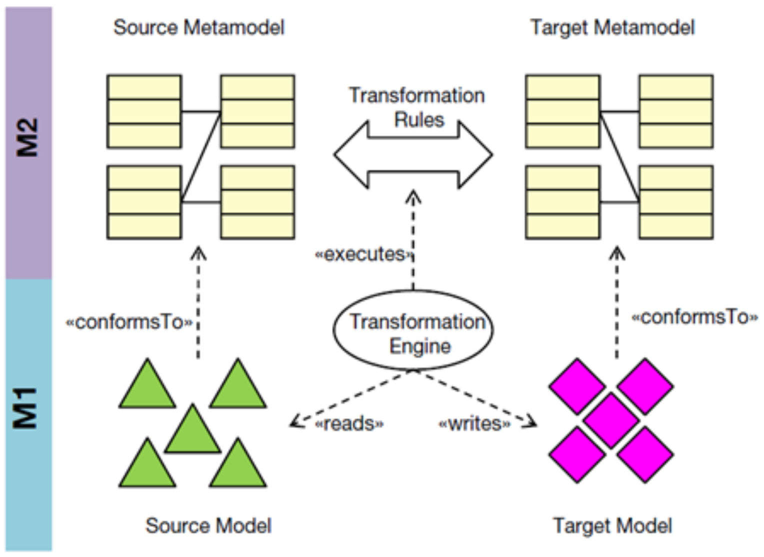


Figure 3.6: Model transformation pattern [Str08]

Query/Views/Transformation (QVT) is a standard set of languages for model transformation defined by the Object Management Group. There are four ways of specifying model transformations (see Figure 3.7). Relations and Core have a declarative style; however, the languages Operational Mapping and Black Box have an imperative style.

The relations language describes the relationship between Meta Object Facility (MOF) models. Also, it records all information that is occurring in the transformation execution by creating trace classes and their instances. This language is more complex than the Core language. Core language treats all elements of source model, target model and trace models symmetrically to evaluate the conditions of pattern matching of a set of models [OMG08].

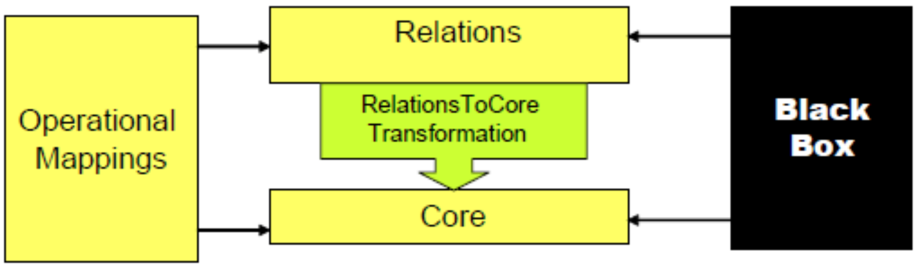


Figure 3.7: Relationships between QVT metamodels [OMG08]

Operational Mapping involves a Relation language with the aim of creating a trace between the elements of model which may be implicit. Also in this syntax we can find imperative constructs, such as loops, conditions or others which are used for instantiating the object patterns specified in the relations. Finally BlackBox implementations explicitly implement a Relation to keep the traces between model elements that are related by the Operation implementation. Only a Relational language is a bidirectional approach for model transformation.

Nowadays tool support for the QVT languages is only in an initial stage because we require time and effort to provide a mature tool, most of which do not encompass all the features of languages. Also, the specification is not officially finalized and it is still unstable. Therefore, the tools are dealing with bug fixes to be used in real industry. In recent years there became available QVT tools which are summarized in Table 3.1.

Table 3.1: QVT Tools per Language [Kur08]

Core	A commercial add-on to OptimalJ
Relational	IKV++ Medini QVT Tata Consultancy ModelMorf MOMENT-QVT Eclipse M2M Relations2ATLVM
Operational Mappings	Borland Together Architect 2006 SmartQVT Eclipse M2M OM2ATLVM

- OptimalJ is an open-source tool which supports the Core language. It was developed by Compuware in 2001.
- For the Relational language there is Medini QVT (among others), an Eclipse-based IDE tool developed by IKV++. Some features of Medini QVT are: syntax highlighting editor, code completion, and debugging facilities.
- Finally, Eclipse M2M supports Relational and Operational language.

ATLAS Transformation Language (ATL) is a textual language which is a mixture of declarative and imperative approaches for model transformation. The tool was developed and maintained by OBEO and AtlanMod in 2003. Basically, ATL aims at valid and executed metamodels, it supports the rules with an imperative or declarative way, and it also supports query, view and transformations. ATL is a unidirectional approach with a read-only source model and it creates a write-only target model. When the tool is executed, only the source model is navigated. A bidirectional transformation in ATL is only possible by creating two transformations, one for each direction. There is a library of ATL transformations from the M2M Eclipse project. The M2M community is growing fast and it contributes to the development of the tool [JABK08].

Graph Grammars (GG) is another declarative and bidirectional approach of model transformation. This method defines that transformations are based on rules that are typically represented schematically. Based on the graphs, the basic idea of Triple Graph Grammars (TGG) is that a graph evolves by applying grammatical rules. Thus, the graph can be separated into three corresponding subgraphs: two subgraphs must obey their own graphic layout and evolve simultaneously, while the third keeps track of the correspondences between the other graphs. A technology that supports TGG is the Fujaba tool suite, an open-source tool created by developers in software engineering. This tool focuses on developing and evolving its features, offering an extensible platform. According to the creators of Fujaba, the tool was originally designed to: "support software in forward and reverse engineering" [Fuj13]. That is why Fujaba is an acronym for "From UML to Java and back again" [Fuj13].

The Epsilon Transformation Language (ETL) is a hybrid model-to-model transformation language that is characterized by dealing with various models of origin and destination. ETL provides scheduling functionality: in this language we can to define lazy rules that are executed only when they are explicitly called and the greedy rules that are executed whenever possible. Rules can be reused and extended through inheritance rule. Epsilon tool, created by Eclipse GMT project, is a recent technology that support the ETL language.

Finally, Extensible Stylesheet Language for Transformation (XSLT) is a functional transformation language for manipulating XML data. Being a functional language, the rules have to be called explicitly and it is strictly unidirectional. XSLT processes a source model with the rules declared in the XSLT; with these instructions, it allows detecting each element of source model and applies the instruction creating an output element.

3.3 Conclusions

This chapter is crucial to know relevant information on the two main themes of the project: BPM and MDD. BPM is an approach that uses MDD practices, so that it initially uses models to represent a business process which will subsequently lead into something executable. Models are the data to be shared among different tools, so model transformation is a very interesting approach that can ensure transparent communication among the systems involved.

The standard notation for modeling business processes aims at setting a universal language so that users of BPM can communicate clearly and easily. However, there are many types of modeling tools that meet the needs of all types of customers, some obey the standard notation for different levels of compliance and others only cover a graphical representation.

The review of the different techniques and technologies that allow transforming one model to another will permit to identify the approaches that can be analyzed in the next chapter. The presentation of a tool by each approach will also allow complementing the analysis in Chapter 4, so you can build a model transformation from a PIM to a bidirectional PSM model.

Finally, after knowing the project key concepts and identifying a strong relationship between BPM and MDE, we can define a methodology to propose a solution to the problem and achieve the project objectives.

Chapter 4

Solution Strategy and Architectural Design

The objective of the fourth chapter is to define a strategy to build a viable solution. Thus, it was necessary to consider the following components: the proprietary modeling notation of Sysnovare, a modeling tool and a model-based transformation approach that allow to communicate with the Sysnovare suite. Defining the strategy of the solution is a key step in the analysis of the problem. Identifying areas of analysis, the advantages and limitations which the BPM suite currently has helped define a proposal.

The BPMN notation can be a useful reference to analyze a proprietary notation, in this case, the Sysnovare Workflow metamodel. There are many modeling tools that support a BPMN notation. The inconsistencies in compliance with the BPMN notation directly affects the interoperability between modeling tools that work in a collaborative environment. Therefore, to select approaches for defining a system that allows transforming a PIM to PSM model are made in order to cover a set of criteria selections that obey the case study requirements.

Finally, after an analysis and selection of the mechanisms that will improve external interoperability on Sysnovare suite, the architectural solution with all the technologies identified in the previous sections is presented.

4.1 Solution Strategy

There are several ways to complement the modeling component of Sysnovare; so it is important to consider all the tools and resources that are present in the case study, i.e. the advantages and strategies of Sysnovare suite and the project objectives; this way, we guarantee to come close to a viable solution.

After analyzing the problem and recognizing the recent research in the context of the project, we believe that using MDD techniques in BPM increments the competitive advantages of the suite.

Thus, to propose a solution to transform the models of business processes we should follow the following steps:

1. To communicate the BPM suite and external modeling tool, a first step is define the concerns and requirements of the approach, in this way easily is defined a set of evaluation criterias for the step 3 and 4.

2. In a model-to-model transformation we must recognize the elements that should be shared in communication. It is important to analyze the proprietary notation of Sysnovare through a gap analysis with respect to the standard BPMN notation. BPMN is a universal notation, so it can be useful to identify if there is partial or full conformity between the notations.

3. Having the knowledge concerning the meta-model of the modeling component of Sysnovare, then it is important to choose a tool or graphical modeling framework that can represent the basic set of elements identified above. This tool must meet the expectations of the business needs in order to select a tool that improves the qualities of the suite. The choice must be selected on based of evaluation criteria that can maintain the approach requirements defined in the step 1.

4. Finally, before we define a solution it is important to evaluate and identify the techniques and tools for the model transformation that can be useful in the composition of the solution. Likewise, in this phase the choices must be selected on based of evaluation criterias that can be maintain the approach requirements defined in the step 1.

4.2 Concerns and Requirements

In this section we define the requirements that will be necessary to select a modeling tool and the best approach for model transformation. Functional and non-functional requirements will be described considering the BPMN elements supported by Sysnovare BPM suite.

Functional requirements:

1. The modeling tool must be able to draw process models with BPMN basic elements: tasks, start event, end event, call sub-process and sequence flow.

2. Application must support team modeling because it is a current feature of the process modeling tool of the suite.

3. Possibility of adding metadata. The modeling tool must support extended attributes for customization because the suite uses specific attributes to model a Business Process (BP).

4. The modeling tool must be able to automatically generate models in XML for importing/-exporting process definitions. Process definitions must be persistent to maintain mapping between the same elements.

5. Communication should be in both directions, bidirectional.

Non-Functional Requirement:

1. Usability: in this item we consider the UI design, learnability and efficiency.
2. Performance: the tool must support up to 10 total users and 2 concurrent users. The number of users is the current quantity of Sysnovare BPM suite users. In group work data will be stored in order, and this characteristic can be completed in a future work in order to support data editing in parallel.
3. Popularity/Maturity: the modeling tool must be in use by a considerable number of users to assure the software evolution in terms of quality and adaptability.
4. Documentation: All information associated to use, manipulation and learnability of the tool.
5. Efficiency: the transformation of models should be quick and consistent in the context of a web environment.
6. Technologic compatibility: As mentioned in Chapter 2, Sysnovare BPM suite is developed in a web environment, so the technologies chosen should be compatible with current technologies in the suite: PL/SQL, HMTL and Javascript.

4.3 Analysis of the Sysnovare Workflow Notation

To facilitate the communication between the suite and modeling tool it is important to specify the model structure of Workflow, i.e. the relational model of the Repository of Process Models (see Figure 2.2).

The Figure 4.1 was created with EMF Project Tool, it depicts a partial metamodel of WF comprising the following concepts:

- **Engine** is a group of models.
- **Model** is a business process model; in this context all processes are private.
- **Activity** is a task that is performed within a process.
- **Transition** describes all information (actions, actors and data) to move from a source activity to a target activity.
- **Subprocess** calls a model and is only used by a type of activity.

Workflow metamodel has more elements which allow to apply all the functionalities of the suite. For this study the partial metamodel is only composed by 5 elements that will be used in the following subsection.

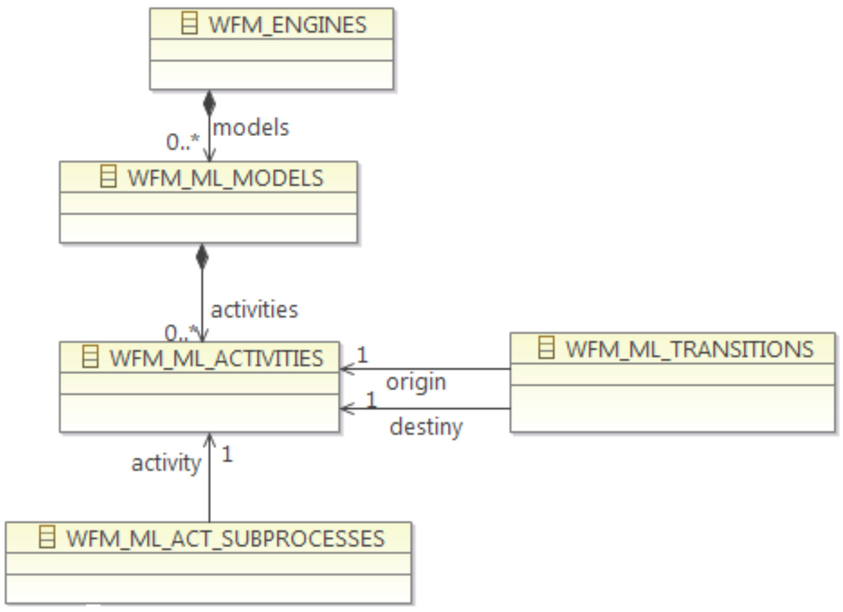


Figure 4.1: Partial Workflow Metamodel (Ecore notation)

Sysnovare Process Modeling Tool obeys a proprietary notation that is misaligned in comparison to notation BPMN 2.0. To achieve an interoperable tool it is important to identify the differences and similarities between notations through a gap analysis. Table 4.1 to 4.6 show the BPMN 2.0 elements supported by Sysnovare BPM Suite, comments and the level of current conformance (CC). In the comments, different types of attributes are defined such as attributes covered, attributes defined indirectly, recommended attributes, required attributes, and missing attributes. Each attribute of BPMN element is mapped with the attribute of workflow metamodel "(WF attribute)".

There are 4 levels for the current conformance of the gap analysis:

- **Level L1** (conceptually) only covers the concept.
- **Level L2** (basic coverage) doesn't cover the attributes that define the functionality of the object, i.e. attributes that define the element behavior are identified in missing attributes or most attributes are defined indirectly.
- **Level L3** (partial coverage) covers the attributes that define the functionality of the object, i.e. attributes that define the element behavior are identified in attributes covered and few attributes are defined indirectly. Also in this level all attributes can be covered but with constant values.
- **Level L4** (total coverage) covers all attributes required and there aren't missing attributes.

Table 4.1: Gap Analysis: Flow elements container

BPMN Elements	WF Elements	Comments	CC
Process (*)	MODELS	Attributes covered: id (code), name (name), documentation (description), isExecutable (always true), processType (always public). Recommended attributes: isClosed, resources.	L3

Table 4.2: Gap Analysis: Flow nodes

BPMN Elements	WF Elements	Comments	CC
Task	ACTIVITIES	Attributes covered: id (code), name (name), documentation (description). Attributes defined indirectly: startQuantity, completionQuantity. Recommended attributes: isForCompensation. The types of activities inherit this attributes.	L4
Service Task	ACTIVITIES / TRANSITIONS	Attributes defined indirectly: implementation, operationRef, dataInputAssociations, dataOutputAssociations, ioSpecification, loopCharacteristics. Missing attributes: boundayEventRefs.	L2
Send Task	ACTIVITIES / TRANSITIONS	Missing attributes: id, name, documentation, messageRef.	L1
Receive Task	ACTIVITIES / TRANSITIONS	Missing attributes: id, name, documentation, messageRef.	L1
User Task	ACTIVITIES / TRANSITONS	Attributes defined indirectly: resources (perfil/structure), ioSpecification, dataInputAssociations, dataOutputAssociations, loopCharacteristics. Missing attributes: renderings, implementation, boundayEventRefs.	L2
Manual Task	ACTIVITIES / TRANSITONS	This type of task inherits the attributes covered of Task.	L4
Call Sub-Process	ACTIVITIES	Attributes covered: id (code), name (name)	L2

Table 4.3: Gap Analysis: Flow node (gateway)

BPMN Elements	**WF Elements**	**Comments**	**CC**
Gateway (exclusive, inclusive, parallel, event-based or complex)	ACTIVITIES / TRANSITIONS	The type of Activity can be OR, i.e., it waits for one transitions or it can be AND, i. e., it waits for all transitions. Also, in a Transition it is possible to define an attribute that specifies a competition, collaboration or specify the accepted transitions quantity. Attributes covered: id (code), name (name), documentation (describe), default for gateway exclusive, gatewayDirection (converging - activity and diverging - transition). Missing attributes for event-based: instantiate - start a process, eventGatewayType - exclusive or only parallel if instantiate is true.	L2

Table 4.4: Gap Analysis: Flow node (event)

BPMN Elements	**WF Elements**	**Comments**	**CC**
Start/End Event	ACTIVITY	It is included in the activity type. Attributes covered: id (code), name (name), documentation (description). Missing attribute: isInterrupting	L2

Table 4.5: Gap Analysis: Flow elements to define a association

BPMN Elements	**WF Elements**	**Comments**	**CC**
Sequence Flow (unconditional, conditional or default)	TRANSITIONS	Attributes covered: id (code), name (name), sourceRef (act_org_id), targetRef (act_dst_id), conditionalExpression (obj_condition), default (obj_action).	L4
Message Flow	TRANSITIONS	Attributes covered: id (code), name (name). Missing attributes: sourceRef (act_org_id + mdl_org_id), targetRef (act_dst_id + mdl_dst_id).	L2

Table 4.6: Gap Analysis: Others elements supported

BPMN Elements	**WF Elements**	**Comments**	**CC**
Data Store Reference	TRANSITIONS	Missing attributes: id, name, dataStoreRef	L1
MultiInstance Activity	TRANSITIONS	Attributes defined indirectly: multiInstanceLoopCharacteristics.	L3
Looping Activity	TRANSITIONS	Attributes defined indirectly: standardLoopCharacteristics.	L3
Standard Loop Characteristis	TRANSITIONS	Attributes covered: id (code), loopCondition (obj_condition).	L4
MultiInstance Loop Characteristics	TRANSITIONS	This class can be parallel or sequence. Attributes covered: id (code), isSequential (always false). Attributes defined indirectly: loopDataInput, inputDataItems.	L3
ResourceRole	TRANSITIONS (Profile/Structure)	Missing attributes: id , resourceRef, resourceAssignmentExpression.	L1
Operation	TRANSITIONS (obj_action)	Missing attributes: id, name, inMessageRef, outMessageRef, errorRefs.	L1

(*) The attributes of elements are optional.

BPMN 2.0 specification [OMG11] describes the attributes of each element.

After performing the gap analysis we can see that the Sysnovare workflow notation supports few elements of BPMN standard notation in its entirety: there are few attributes that are covered and for some elements it is not possible to define a relationship of one to one. Consequently, we define pairs of basic elements that can be used in communication (see Table 4.7):

Table 4.7: Mapping elements

BPMN element	WF element
Process	WFM_MDL_MODEL
Task	WFM_MDL_ACTIVITIES
Start Event	WFM_MDL_ACTIVITIES
End Event	WFM_MDL_ACTIVITIES
Sequence Flow	WFM_MDL_TRANSITIONS
Call Sub-Process	WFM_MDL_ACTIVITIES

4.4 Selection of a Graphical Modeling Framework

This research attempts to find modeling tools of different vendors which partially or completely support BPMN. Table 4.8 shows the selection criteria with their description and weights (W). The weight was assigned according to the requirements defined previously, the importance and priority of each criterion.

Table 4.8: Modeling Tool: Selection criteria and relative weights

Evaluation Criteria	**Description**	**W**
Popularity / Maturity	The tool must be used by a considerable quantity of users or companies to assure the software evolution in terms of quality and adaptability.	3
Extensibility / Customization	Create new elements, and add attributes to basic elements. Customization of elements in terms of design and data properties.	3
Modeling	Drawing features such as drag-and-drop, easy connections between objects and edition of object properties.	3
Usability	Tool features such as UI design, learnability and efficiency.	3
Export / import persistent models in XML	Ability to export and import models in XML format. The elements of the model should have a persistent identification, to allow the mapping between the same model elements designed in modeling tool and model stored in the suite Sysnovare.	3
Team modeling	Possibility to share a model allowing collaboration in the design phase.	2
Standard Compliance / Interoperability	Compliance with the standard notation allowing interoperation with other modeling tools with the same level of compliance.	1
Documentation and Tutorial	All information associated to use, manipulation and learnability of the tool.	1
Extras	Tools extra features.	1

Evaluation criteria with more weight such as modeling, extensibility, usability, popularity/maturity and export/import persistent in XML models are used for selection and creation of a short list of modeling tools, which will be analyzed in more detail.

Four modeling tools obey the criteria with a higher weight. The short list includes TIBCO Business Studio [TIB13], BMPN2 Modeler [The13b], Draw.io [JGr13a] and Bizagi Modeler [Biz13]:

1. Bizagi Modeler is a desktop application developed by Bizagi and it is a component of BPM suite. The tool is a freeware application which not only allows modeling the processes conforming to the standard notation but also allows creating process documentation in different formats and performing process simulation to analyze its behavior.

2. TIBCO business studio is a free edition tool created by TIBCO and it is part of TIBCO software that supports business process management. It is an Eclipse-based IDE tool, allowing more flexibility and with the capacity to communicate with other tools, as opposed to Bizagi. This tool does not support all the elements of BPMN 2.0. However, it is extensible because it allows adding attributes to the elements.

3. BPMN2 Modeler is a project of The Eclipse Foundation. It allows a personalization through plugins; it is one of the tools with greater extensibility as opposed to the previous two. Currently, it supports all the standard notation elements.

4. Draw.io is an online diagramming application that allows you to draw the following diagrams: flowcharts, UML, Entity-relationship diagrams, networks, models of business processes (BP), organization charts, electronic circuits, wireframes and prototypes. It uses the drag & drop technique and it is a simple tool with high level of usability. Mainly, it is very flexible allowing the personalization of graphic elements.

In Table 4.9 we can see that we have tools for almost all operating systems and these can be used freely. Table 4.10 illustrates the results of each criterion for the four tools. Most of the tools support a basic compliance with the BPMN notation, i.e., compliance with the basic elements such as process, tasks, simple events (start, end), gateways and sequence flows.

Table 4.9: Modeling Tool: Information about tools in the short list

Tool / **Criteria**	**Bizagi Process Modeler**	**Draw.io**	**TBICO Business Studio**	**BPMN2 Modeler**
Creator	Bizagi	JGraph	TIBCO	The Eclipse Foundation
Platform / OS	Window	Web application	Window and Linux	Eclipse-based IDE
Version of BPMN	2.0	Not specified, can be implemented by the user	1.2	2.0
Software Licence	Freeware	Open-source	Freeware	Open-source

Table 4.10: Modeling Tool: Evaluation of the tools in the short list

Tool / **Criteria**	**Bizagi Process Modeler**	**Draw.io**	**TBICO Business Studio**	**BPMN2 Modeler**
Popularity / Maturity	It is used by many users in the BPM domain. Recently updated.	It is used for modeling and it can evolve with customization	It is used by many users in the BPM domain. Updated in 2012	It is a project that can evolve with customization. Used by Eclipse community.
Extensibility / Customization	It allows adding attributes of elements and creating specific types of elements	The tool allows high customization because it is possible to create elements, adding attributes in terms of design and information with simple plugin development in javascript. Also is possible to define a new format to export / import models	It allows adding attributes to elements.	It allows creating elements, adding attributes and customizing the design and other information by building plugins in Java.
Modeling (Drawing features)	All features describe in Table 4.8	All drawing features	All drawing features	All drawing features
Usability	Intuitive and easy to use for common users	Intuitive and easy to use for common users	Intuitive and easy to use Eclipse based applications	Intuitive and easy to use for users that use Eclipse based applications
Export / import models in XML	XPDL, BPMN, Visio, Image (only in export) - Persistent	XML - Persistent	XPDL - Persistent	BPMN - Persistent
Team modeling	Yes	Yes	No	No

Standard compliance / Interoperability	Support almost all elements and it can interoperate with other tools that support XPDL 2.2	Only support the elements implemented by user and it can interoperate with another tool in any format if this feature is implemented.	Supports some elements and it can interoperate with another tool that supports XPDL 2.1	Supports all elements and it can interoperate with more tools because of export / import in BPMN format in the last version.
Documentation and Tutorial	More information to learn BPMN	More information technique	More information technique	More information to learn BPMN
Extras	Generating documentation, simulation	Customization through simple plugins in javascript	Organization model	Workflow patterns

In Table A.1 in Appendix A we sumarize the evaluation results.

All tools are easy to use and very intuitive, some have more information than others, often depending on whether the tool is part of a commercial product or if is a free project of collaboration.

Draw.io is a more extensible solution which allows adapting better to the necessities of organizations; however, it does not have a defined compliance because it can be constructed according to the elements that users wish to be supported, depending on the domain of the business. Another important aspect is the possibility for tools to interoperate with other modeling tools through import/export of XML files. The tool is a specific application for modeling and therefore does not belong to any BPM suite that supports this feature. Draw.io is an example of the products of JGraph [JGr13b], which has a JavaScript HTML 5 Diagramming Library: mxGraph; you can easily create and integrate a modeling tool in a web environment.

Bizagi and TIBCO, being part of a potential product considered in the BPM market, not only provide information for modeling but also knowledge and help in the learning of standard notation use. The other tools have more technical information on how to customize the tools.

Finally, as Sysnovare prefers to integrate a solution that can further be customized and be profitable in other areas, the selected tool is Draw.io. This tool will be adapted to the needs of the company, thus the company chose to buy the Graphics engine JGraph: mxGraph.

4.5 Selection of Model Transformation Technology

To develop a model transformation solution it is important to try and test the technologies and techniques that can be adapted and used in our context. Table 4.11 shows the selection criteria, description of the criteria and their weight (W) for evaluation. Again the weight of the criteria has been assigned according to the conditions defined in Section 4.2, the importance and priority of each criterion.

Table 4.11: Model Transformation Technology (language and tool): Selection criteria and relative weights

Evaluation Criteria	**Description**	**W**
Popularity / Maturity	The transformation language must be maintained by a considerable group of researchers or organizations. A more mature language processing can be useful to define complete and stable approaches.	3
Integration with other technologies	It is important to consider whether the tool is compatible with the technologies of BPM Suite; this way, the integration does not need to examine alternative programming languages that are compatible with a PL/SQL. For example, Java language is completly supported by Oracle and it allows easily embed Java code in PL/SQL.	3
Expressive Power	The transformation language can be difficult to understand the syntax. For a programmer, language should be perceptible so that creating a code may be an easy process.	3
Maintainability	Considering that the approach can go through a process of improvement and optimization, it is important to analyze the effort required to: redefine the metamodels, make changes in mapping rules and integrate new features.	3
Level of abstraction	The context of the project is to transform a PIM to a PSM model. Thus, an important factor to define the approach is the amount of changes needed to reach the PSM or PIM model in the case of obtaining the backward transformation.	2
Coverage of Communication	To establish a communication in both directions, the approach must be bidirectional. However, to consider a unidirectional option is not a feature to limit the selection of the approach. If a technique is unidirectional, it simply requires two transformations to be defined.	1
Usability	The ease with which the user can manipulate the tool will be considered in the criterion. Thus, the characteristics of compression and operability are taken into account to evaluate the tool supporting a given language transformation.	1

The selection criteria allow categorizing transformation techniques in the context of the project. The analysis of transformation languages through the tools that support them allows a previous evaluation of the transformation component of models to be implemented in the BPM suite. Creating an efficient, flexible and capable of evolving mechanism requires the definition of a transformation of a PIM to PSM model through a robust and intuitive language. This way, the criteria that prevail throughout the assessment are: ease of maintenaince, expressive power of the notation, ease of integration with BPM suite and the maturity of the language together with the tool that supports it.

After presenting a group of approaches in Chapter 3 and considering the evaluation criteria described above, 4 model transformation tools were identified and selected for further analysis. An overview of the tools and their transformation approach follows:

1. QVT-R supported by Medini QVT: Medini QVT [Ikv13] is a model transformation tool that supports transformations based on a relational language. This tool provides a plug-in to integrate applications in Java; using the plug-in it is possible to support transformations expressed in QVT-R. The tool works as a transformation engine; it basically needs models in XMI format, metamodels in Ecore format and transformation rules in QVT-R language.

2. ATL supported by the M2M Eclipse project [The13a]: This tool developed on the Eclipse platform acts as a transformation engine like Medini QVT. ATL works with models in XMI format, Ecore metamodels and transformation rules in ATL format. The transformation engine can also be incorporated into Java applications.

3. XSLT supported by XMLType API: XMLType API for PL/SQL [Ora13] supports the transformation of XML files based on the rules defined in XSLT format. To perform the transformation, as a first step you need to create an XML file using a function that gets the file contents. Then, from the XML file one executes the transformation function, which receives as parameter the mapping rules in XSLT format.

4. TGG supported by Fujaba: it is a more dynamic approach that creates mapping rules using a visual model based on triple graph grammar. This approach applies transformation rules incrementally. Through the Fujaba [Fuj13] interface you can set the rules and the metamodel through graphic elements:

   - The source model is defined on the left side of the graph;
   - A rule is defined by a node; the graphic element has associations with two elements to signal a relationship;
   - Finally, the target model is modeled on the right side.

Having an overview of the approaches, the results of each evaluation criterion by methodology are presented in Table 4.13:

Table 4.12: Model Transformation: Information about approaches

M2M approach / Criteria	**QVT-R supported by Medini QVT**	**ATL supported by M2M Eclipse**	**XSLT supported by XMLType API**	**TGG supported by Fujaba**
Creator	ikv++ technologies ag	The Eclipse Foundation	Oracle	Fujaba development group
Platform / OS	Eclipse-based IDE	Eclipse-based IDE	Java API	Eclipse-based IDE
Software Licence	freely available	open-source	freely available	open-source

Table 4.13: Model Transformation: Evaluation of the approaches

M2M approach / Criteria	**QVT-R supported by Medini QVT**	**ATL supported by M2M Eclipse**	**XSLT supported by XMLType API**	**TGG supported by Fujaba**
Popularity / Maturity	The tool has been evaluated in several scientific articles and case studies. Currently, it supports the functionalities of QVT-R.	It presents a higher maturity than QVT-R, but it is an older technology.	For the purpose of the project, it is robust in the support of XSLT language.	The tool is studied and evaluated in approaches with incremental changes. It presents robustness in the support of language.
Integration with other technologies	Plug-in for integration in Java applications.	Plug-in for Java applications.	Compatible with PL/SQL.	Code generation in Java.
Expressive Power	QVT-R language is declarative, consistent and intuitive.	ATL is also a hybrid, perceptible and easy to learn language.	XSLT language has a complex syntax.	TGG has a robust syntax. However, it is intuitive, as it defines rules through a graphical display.

Maintainability	For the maintainability of the metamodel, the tool provides an edit mode; you can equally update mapping rules.	Changes in metamodels are done through the edit mode. One can easily add a class, attribute or association in the metamodel. However, transformation rules require greater effort if there is a bidirectional transformation.	To keep the transformation rules one only needs to change the XSLT data.	Changes can be performed through a GUI. In this case, one must make the changes through the Fujaba editor. The other tools allow you to edit the rules in any text editor.
Level of Abstraction	It takes two transformations to get a PSM model from PIM. The first transformation yields the model in XMI format.	Through two transformations it is possible to obtain a PIM model for PSM. Similarly to the previous approach, a transformation is necessary to obtain the model in XMI format.	A transformation is enough to get a result.	A transformation is necessary to obtain a PSM model from PIM. Yet, another transformation is needed for the tool to interpret the entry model.
Coverage of Communication	Bidirectional	Unidirectional	Unidirectional	Bidirectional
Usability	The tool is intuitive and simple to use. Based on Eclipse-IDE, through the execution of a command it allows you to perform the transformation.	Based on Eclipse-IDE, the tool has the same features as the Medini QVT: Intuitive and easy to use.	As it is not a tool, this criterion is not evaluated.	Fujaba suite features an intuitive graphical environment; through a graphical interface, it is possible to create mappings between elements quickly and easily.

In Table A.2 in Appendix A we sumarize the evaluation results.

XSLT language has a complex syntax and the model transformation requires a more elaborate mapping with composite associations; as the complexity of the syntax is more useful for the reorganization of data, consequently, this approach is not the best option for models transformation.

Moreover, Medini QVT and QVT-R language, define an interesting approach to the project. The tool has features that facilitate the integration with the BPM suite. Through a plug-in, the functionalities of the transformation engine can be integrated into an application developed in Java. Besides, QVT-R is an intuitive language and with a simple syntax in comparison with XSLT.

The approach of ATL language is consistent with the purpose of the project and it is a language with more maturity than QVT-R. However, ATL remains as a second choice to perform model transformation since ATL is unidirectional and QVT-R language has a reasonable maturity. Taking into account the maintenance of the approach, changing two rules in comparison to one can be an important factor to consider. The number of rules can be high depending on the evolution of the number of modeling elements to support.

The models generated by the tools of the case study are defined in an XML format. Since two of the approaches of transformation models require models in XMI format, we can identify a basic transformation among XML files. In this case, XMLType API can be considered as the best approach. At this level of data processing, transformation in XSLT can be useful to re-organize data and get XML that conforms to a specific format, XMI. Besides, in terms of performance this API can contribute to the efficiency of the system.

Supported by Fujaba TGG is another interesting and more complete approach in comparison with the two ones considered above. This approach not only has a more dynamic way of defining the transformation rules, as it is also considered a powerful tool because it supports incremental synchronization. Fujaba also generates Java code; this way, it allows integration with other libraries. Just as there are advantages to the tool, there are also limitations. A limitation of the tool is not to allow the exchange of models through XMI files. Thus, implementing this component can take extra time to establish communication with the suite.

Finally, in more than one approach it is necessary to define the metamodels in Ecore format. As an auxiliary tool, the Eclipse Modeling Framework Project (EMF) [The13c] can be used to create metamodels in a visual mode and dynamic way.

4.6 Proposed Architecture

In the following section we describe the solution proposed to the Sysnovare problem presented in the first chapters. The problem analysis and tool selection that allow to test and subsequently to develop the approach were key elements to define an automated system that allows interoperating two systems: Sysnovare BPM suite and the Graphical Editor (based on Draw.io). Given that both tools have the possibility to import and export the models through XML files, the transformation component of models will use the mentioned files in XML format as input and output data.

The proposed solution is intended to complete the suite modeling component. For the mentioned purpose the component must integrate the graphics engine mxGraph of JGraph. The diagramming library mxGraph allows development of graphical editor built on based of Draw.io tool. Since the mxGraph library is developed in javascript, it can be embedded in the Sysnovare suite. In Figure 4.2 we can see a view of the tools integration.

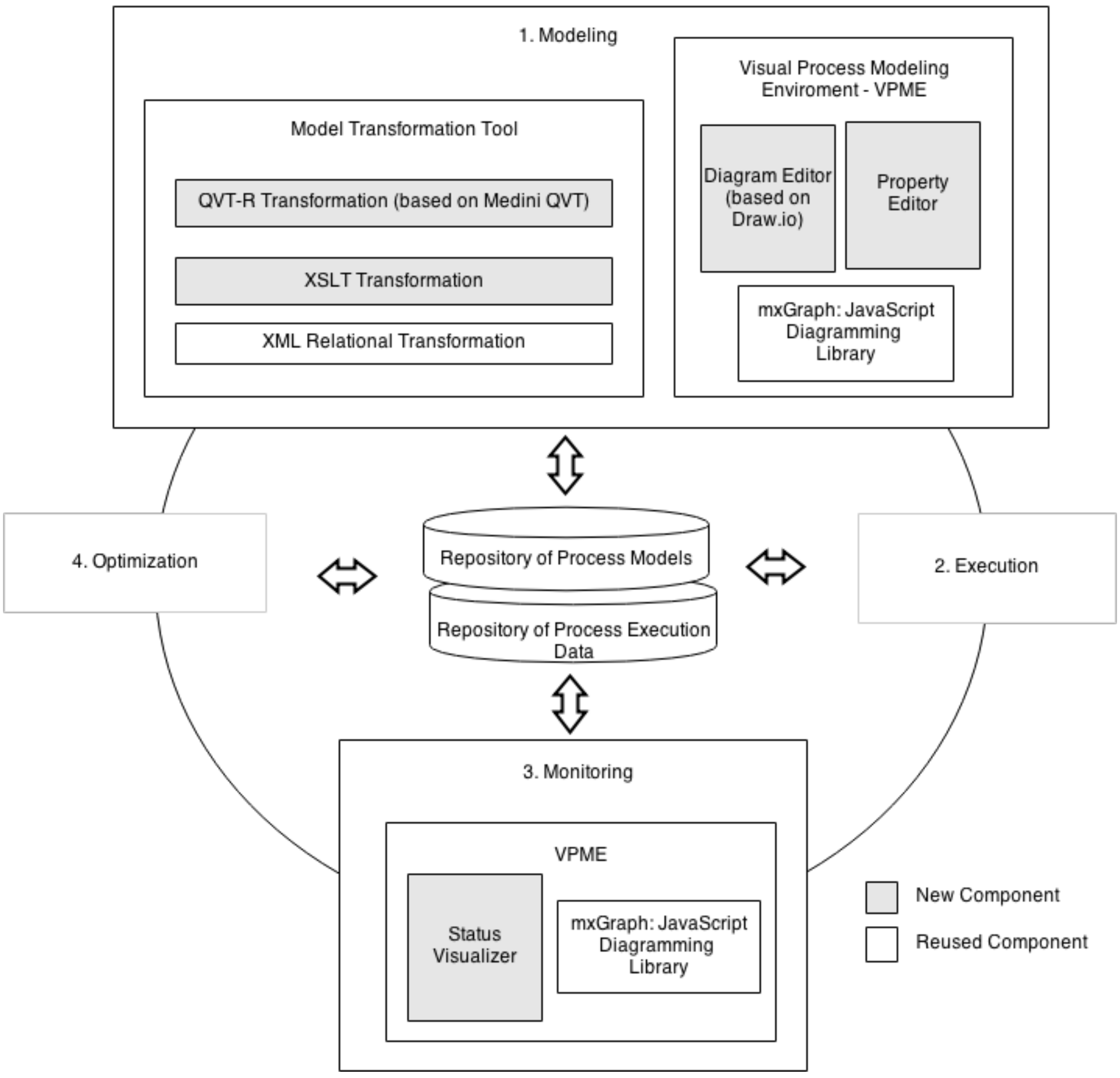

1. Modeling
Model Transformation Tool
QVT-R Transformation (based on Medini QVT)
XSLT Transformation
XML Relational Transformation
Visual Process Modeling Enviroment - VPME
Diagram Editor (based on Draw.io)
Property Editor
mxGraph: JavaScript Diagramming Library
4. Optimization
Repository of Process Models
Repository of Process Execution Data
2. Execution
3. Monitoring
VPME
Status Visualizer
mxGraph: JavaScript Diagramming Library
New Component
Reused Component

Figure 4.2: A proposed architecture for integrating tools

Considering that the proposed approach will directly affect the modeling and monitoring components of the suite, the graphical editor can have more than one mode of viewing and editing. In the Figure 4.2 we can see there are 3 modes that composes the visual process modeling enviroment. Following the short description of each mode:

- **Diagram Editor** is developed on base of Draw.io. This mode aims to build the model design, i.e., insert and relate the graphical elements.
- **Property Editor** is a second mode that aims to edit the element attributes. In this mode some alterations influence the appearance of the graphical element.
- **Status Visualizer** is a last mode used for visualise the informations of current state of a process instance. In this case it will be more widely used in a phase of control and not of modeling. A third mode will not need to perform transformations between models.

Regarding the communication mechanism between tools, the Model Transformation Tool will be invoked automatically through the diagram or property editor. The synchronization utility (XML Relational Transformation) developed by Sysnovare will be used to transform a XML file into PL/SQL code for populating the repositories of models. To perform the transformation of XMLs, the XSLT Transformation will be developed to convert XML files between the formats understood by the tools and the XMI format used by the next component. Finally, the QVT-R Transformation based on Medini QVT tool will transform a model of the BPM suite to model of the Graphical Editor. A bidirectional transformation will be possible through the last component that supports the QVT-Relational notation.

4.7 Conclusion

In this chapter we identify the limitations and requirements depending on the technologies involved in the approach; these variables direct the proposed solution. The analysis of the misalignment to the standard BPMN notation evaluates and identifies a set of basic elements to communicate among systems: Task, Sequence Flow, Start Event among others. Through a comparative analysis, we check whether the Sysnovare metamodel covers a basic set of BPMN elements. Since compliance to the standard notation is basic, the result may not hinder, in a future project, the possibility to evolve the approach to map more BPMN elements in the BPM suite.

There are many modeling tools for all types of users in the area of BPM. Since interoperability between tools is still an area under study, the best option for Sysnovare is to integrate Draw.io tool, a technology based on a graphic library specialized in modeling technology. The tool enables customization of the application in Sysnovare solutions, thus the competitive advantages of the product in the BPM market is maintained.

The MDD approaches will be interesting features so that more than one technology is chosen to integrate the processing solution. QVT-R and XSLT are the languages used in two levels of templates that the proposed transformation will define. Selecting an only approach does not mean covering all model levels that can exist in a system. In the case more than one technology are conjugated to develop the transformation model tool.

To close this chapter, after checking the resources of the case study and the technologies that could integrate the solution, a proposal based on the conclusions that were obtained in each section was presented. Auxiliary tools identified in the analysis and selection of technologies will be useful to assess the proposal in general terms.

Chapter 5

Visual Process Modeling Environment

This chapter focuses on describing the visual layer of process modeling. Given that this layer runs in a web environment, this chapter will present all graphical components that allow the user to interact with the modeling component and the internal mechanism that performs model transformation. Visual Process Modeling Environment is the medium that allows the end user to create or edit process models for execution with the Sysnovare BPM suite, or, on the contrary, obtain a diagram of a model from a business process already created in BPM suite.

In order to maintain the competitive advantages of BPM suite, an adapted version of Draw.io will be implemented after acquiring and integrating the graphic engine mxGraph into BPM suite. With the framework mxGraph and adapted version of the modeling tool, we obtain a simple tool with high usability, flexible and adaptable to the BPMN notation.

For development of the visual process modeling enviroment we will need to create BPMN elements and introduce three edit modes to use the graphical editor. Therefore, we will adapt the integration of a modeling tool to the managing mechanisms of Sysnovare business process.

5.1 Integration of framework: mxGraph

To embed the graphical editor in the BPM suite, one first needs to integrate the graphics engine mxGraph. Integrating a graphics library in javascript in a web environment is a simple step. Since the suite is developed in a web environment, one only needs to store in the system mxGraph JavaScript client, which should be loaded automatically after invoking the web page that gives access to the Graph Editor. According to JGraph: "This is an incredibly simple architecture that only requires a web server capable of serving html pages and a JavaScript enabled web browser" [JGr13b].

BPM suite is a web application developed in PL/SQL, as mentioned in Chapter 2; to implement the graphical interfaces it will be necessary to create procedures in PL/SQL. Such procedures will create the "View" layer invoking the classes that constitute the adapted version of Draw.io.

5.2 Diagram Editor

The first mode of the Graphical Editor will allow modeling business processes. Thus, BPMN elements can be put in the tool only in a graphical representation, relate the mentioned elements and edit the attributes of each graphical object. For the user of the BPM suite to become familiar with the BPMN notation, the names of the graphical objects of the editor will be displayed with the names of the elements of BPM suite (see Figure 5.1). This way, the mapping of names of graphical objects is as follows:

- Task - “Atividade”
- Start Event - “Atividade Inicial”
- End Event - “Atividade Final”
- Call Sub-Process - “Sub Processo”
- Sequence Flow - “Transição”

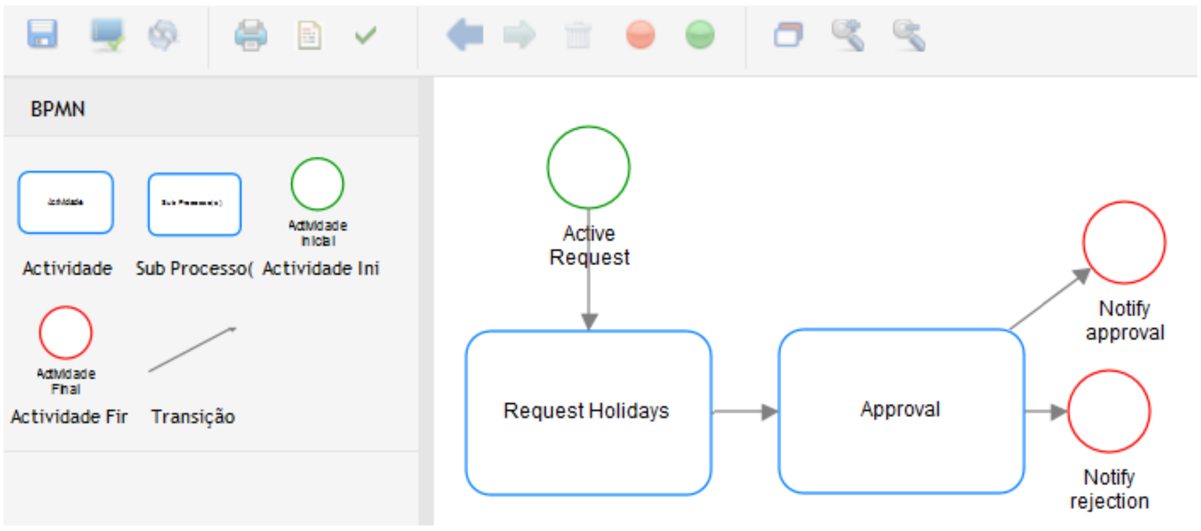


Figure 5.1: Diagram Editor (in portuguese)

To implement each BPMN element it was necessary to use mxGraph classes and create 5 templates in "BPMN" sidebar (see Figure 5.1). Since the mxGraph library is complete, one can define the elements by defining the graphic features (see Listing 5.1), and specify their attributes.

```
Sidebar.prototype.addBpmnPalette = function(dir, expand)
{
  this.addPalette('bpmn', 'BPMN ', true, mxUtils.bind(this, function(content)
  {
    content.appendChild(this.createVertexTemplate('shape=ext;rounded=1;strokeColor
        =#007FFF', 120, 80, 'Task', 'Actividade', true));

    content.appendChild(this.createVertexTemplate('shape=ext;rounded=1;strokeWidth
        =3;strokeColor=#007FFF', 120, 80, 'CallSubProcess', 'Sub Processo(s)', true
        ));

    content.appendChild(this.createVertexTemplate('ellipse;verticalLabelPosition=
        bottom;verticalAlign=top;perimeter=ellipsePerimeter;outline=standard;symbol
        =general;strokeColor=#009900', 40, 40, 'StartEvent', 'Actividade Inicial',
        true));

    content.appendChild(this.createVertexTemplate('ellipse;verticalLabelPosition=
        bottom;verticalAlign=top;perimeter=ellipsePerimeter;outline=end;symbol=
        general;strokeColor=#FF0000', 40, 40, 'EndEvent', 'Actividade Final', true)
        );

    content.appendChild(this.createEdgeTemplate('endArrow=block;endFill=1;endSize
        =6;verticalAlign=top',200, 100, 'SequenceFlow', 'Transicao', true));

    }));
}
```

Listing 5.1: Adding BPMN element templates

Finally, as Draw.io is a complete tool with several features, it was necessary to choose a basic set of actions. The actions implemented in the editor are described below:

1. **Save**: to store the model it was necessary to use the HTTP request method. Basically to perform this action, the tool communicates with the server by invoking a procedure in PL/SQL; so the mentioned procedure receives the contents of the XML file and then stores the data in the database. Thus, each time the editor is used we can see the current state of the graph.

2. **Publish**: to publish a model, it is first necessary to store the diagram, i.e. automatically invoke the "Save" action and then invoke the model transformation procedure in the direction mxGraph -> Sysnovare.

3. **Refresh**: for the user to view the current diagram, it was necessary to place an order to the database through the HTTP request method. This action allows getting the latest data from the XML file that stores the diagram of the model.

4. **Validate**: validation was implemented to ensure consistent processing models without lack of data or ill formed attributes.

5. **Active/Inactive**: in Sysnovare BPM suite, in addition to the removal of elements, it is possible to activate/inactivate objects. Since this is a feature that allows consistency in data synchronization, it was required to create the two actions in the modeling tool. For these two actions the tool makes the graphic as visible or invisible.

5.3 Property Editor

The second mode of the graphical editor allows the user to edit the properties of graphical objects. In this mode you cannot add or change relationships between elements. The separation of this interface with Diagram Editor allows the user to divide the work and save space in the database; as in the first mode you do not need the graphical objects stored in the Repository of Process Model (RPM) of BPM suite.

To edit the properties of each element of the model it is required that the process model has been published in Diagram Editor, i.e. to run the model transformation, and as a result, store the templates in the repositories of BPM suite. Thus, Property Editor can use the XML file generated by the modeling tool and create links between graphical objects and the elements stored in RPM (see Figure 5.2).

If the model has not been published, the editor allows you to view it but it does not allow editing the data elements. As the data changes are directly affected in RPM, if the element is not found, the interface displays a message indicating the problem.

The second mode allows performing the following actions:

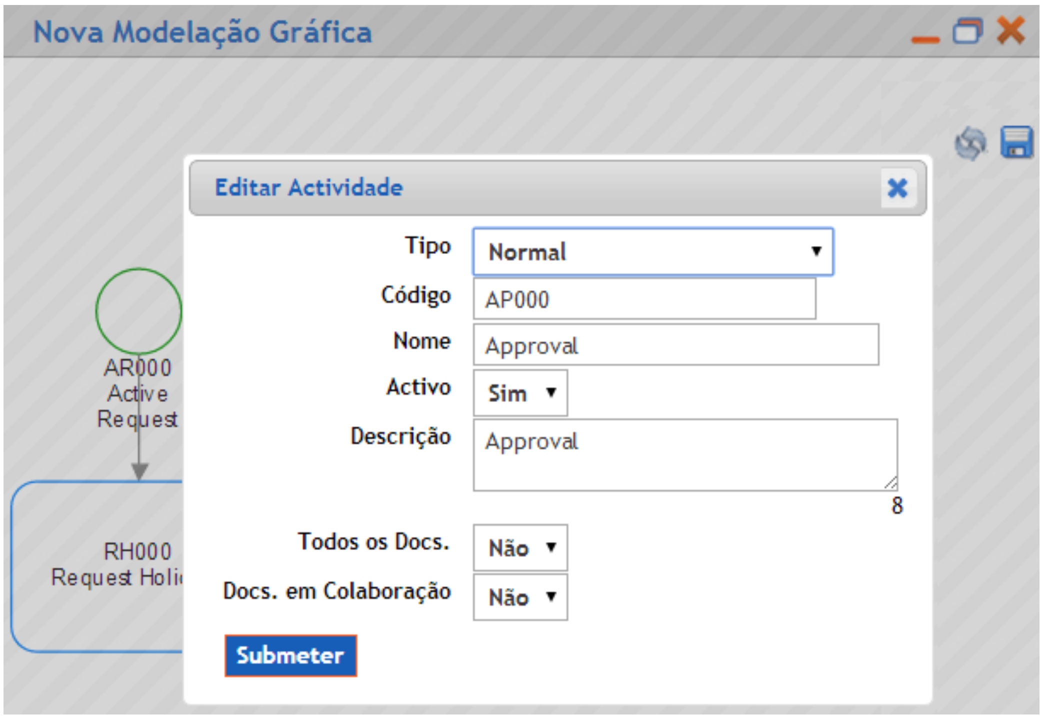
1. **Refresh**: this action makes a request to the database for XML diagram of the latest model.

2. **Save**: as it is possible to modify the attributes of graphic elements directly in RPM, the "Save" action allows the user to update data changed in the XML editor file. Basically, this action changes Sysnovare Workflow Model to mxGraph model, ensuring to keep the data updated. As the editor of some attributes can generate graphical differences, i.e. shapes and colors or relationships between objects, data updating allows the user to work with consistent data.

5.4 Status Visualizer

The last mode of the Graphical Editor will be used strictly for the monitoring component of the Sysnovare suite. The features of mxGraph framework allow you to change the state of the graphical objects based on a predefined diagram, in this case XML created by the Diagram Editor. In Figure 5.3, we can see an example of a business process that was modeled and executed.

In the mode of Status Visualizer you can view the information of the current state of each graphic object; the user can see the current state of the workflow in real time. At the interface,



Tipo	Normal
Código	AP000
Nome	Approval
Activo	Sim
Descrição	Approval
	8
Todos os Docs.	Não
Docs. em Colaboração	Não

Figure 5.2: Property Editor (in portuguese)

through the icon "Refresh" you can see if the model behavior has been recently changed. To perform such action the editor queries the information of each element in the repository of execution model; this way, the editor processes the information and updates the graphic characteristics of the element.

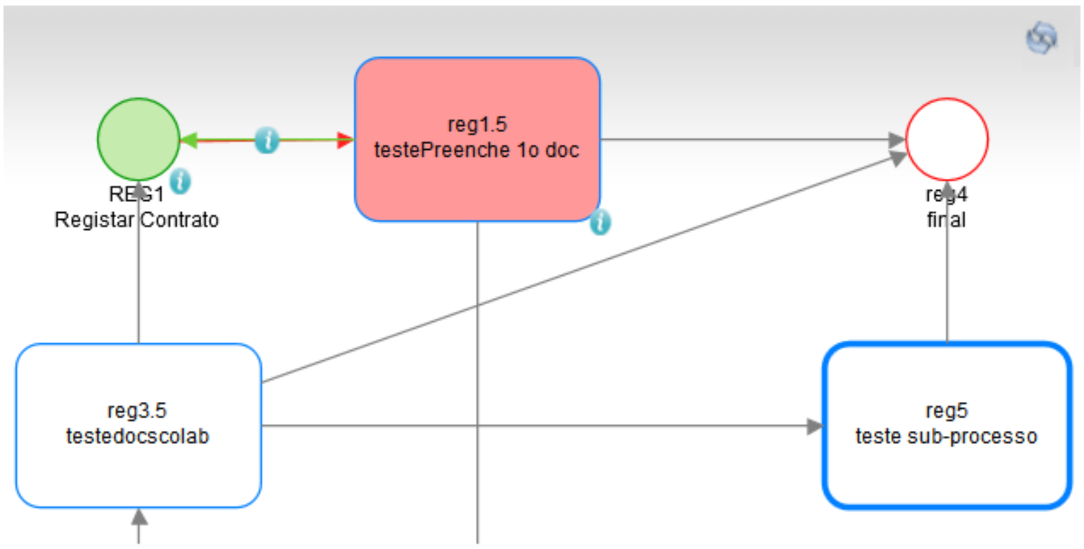


Figure 5.3: Status Visualizer (in portuguese)

5.5 Conclusion

The visual environment was not only useful for the modeling component, as it was also possible to contribute in the graphical environment of the monitoring component. Based on the choice of the process modeling tool, it was possible to adapt the tool to the objectives of Sysnovare, and therefore get the most benefit for the two components of the BPM suite.

In this chapter we can see that the requirements specified in Section 4.2 were met. Thus, the modeling environment allows the user to model a process with the basic set of BPMN elements in a simple and intuitive tool, which allows working in a dynamic environment.

This layer can be improved and easily adjusted if necessary, by adding other features that enable to manage modeling in a better way. An example of this will be to choose the graphic type to be created after inserting a connection from a source element.

Chapter 6

Model Transformation Tool

The sixth chapter describes the work performed on the model transformation layer. This layer is responsible for establishing communication between the tools involved in the system, i.e., the modeling component of the Sysnovare suite and the previously presented layer: Visual Process Modeling Environment based on mxGraph framework.

The techniques and tools chosen in the Chapter 4 were selected in order to automate model transformation, obtaining an efficient response for users to view the results in the graphic layer. The languages that have been chosen to develop these layers are Extensible Stylesheet Language Transformation (XSLT) for the layer of data processing and Query/View/Transformation Relational (QVT-R) for the second layer of model processing.

To ensure efficiency in user queries, the data processing layer is developed with XMLType API for PL/SQL, automatically performing a transformation that obeys the rules defined in XSLT. The first layer is responsible for preparing the data in a XML Metadata Interchange (XMI) format to specify the entry model; later, after receiving the result of processing, the layer processes the data output in XMI format to convert them to an XML file.

The second layer is represented by a newly developed Java library that provides an API for transformation and writing of the models in the operating system. Finally, the graphical layer receives the model in a XML file for rendering the diagram. API transformation includes a plugin that supports the QVT-R language provided by the Medini QVT tool. In the Medini QVT modeling tool, metamodels are defined in Ecore format, transformation rules are defined in the QVT-R language and finally the models are instances of the metamodel and represented in XMI format.

The transformation layer is implemented with technologies and approaches that can be replaced or that enable the component to evolve with more sophisticated features. An example for further work in this component will be the possibility of performing incremental changes in an environment with model updates in parallel.

6.1 Model Transformation Flows

Before presenting all the components that make up the developed model transformation solution in Figure 6.1 we can see the flows of the internal functioning of the system.

Then, the forward transformation is described as follows:

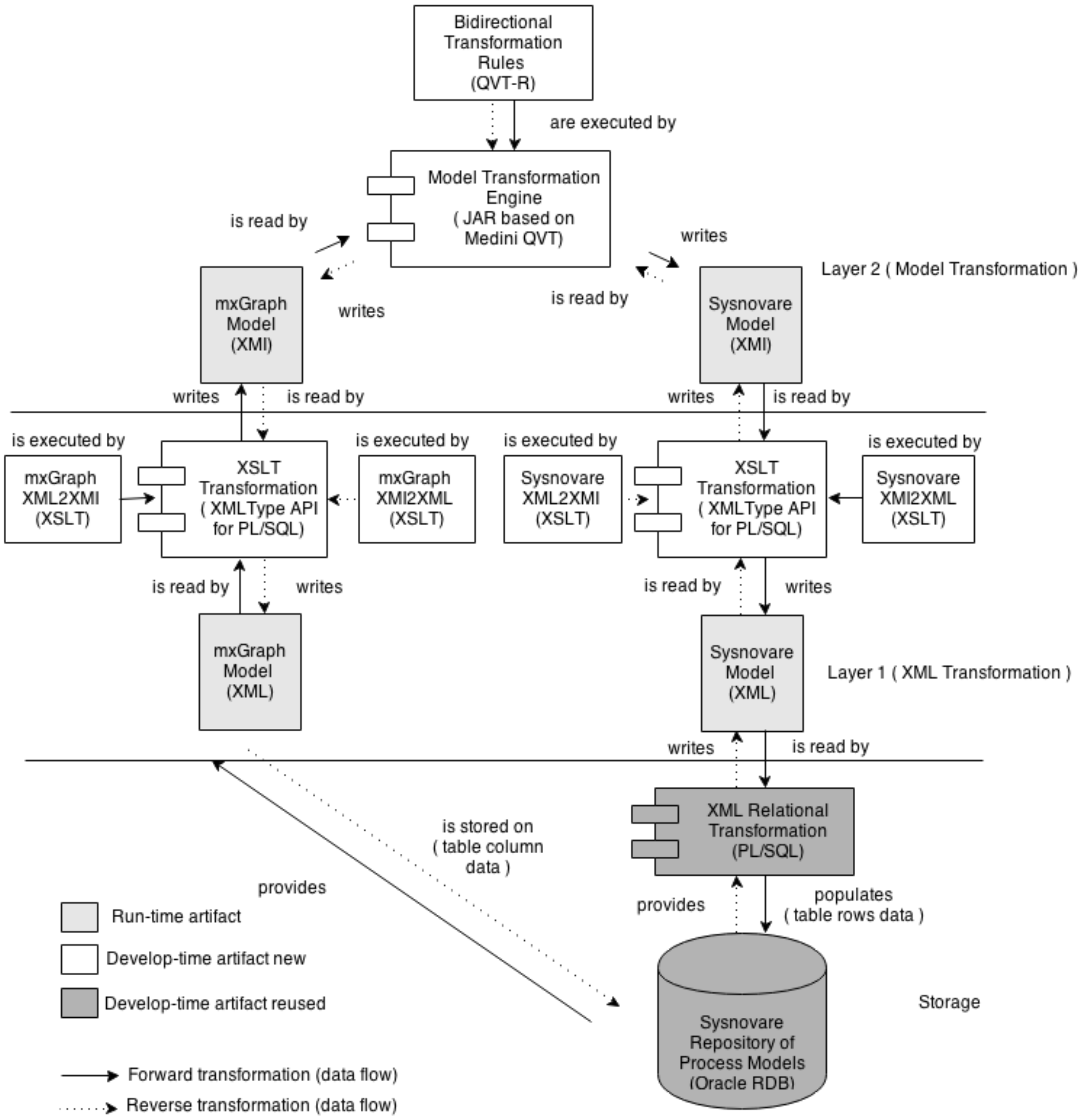

Bidirectional Transformation Rules (QVT-R)
are executed by
Model Transformation Engine (JAR based on Medini QVT)
is read by
writes
writes
is read by
Layer 2 (Model Transformation)
mxGraph Model (XMI)
Sysnovare Model (XMI)
writes
is read by
writes
is read by
is executed by
is executed by
is executed by
is executed by
mxGraph XML2XMI (XSLT)
XSLT Transformation (XMLType API for PL/SQL)
mxGraph XMI2XML (XSLT)
Sysnovare XML2XMI (XSLT)
XSLT Transformation (XMLType API for PL/SQL)
Sysnovare XMI2XML (XSLT)
is read by
writes
is read by
writes
mxGraph Model (XML)
Sysnovare Model (XML)
Layer 1 (XML Transformation)
writes
is read by
XML Relational Transformation (PL/SQL)
is stored on (table column data)
provides
provides
populates (table rows data)
Run-time artifact
Develop-time artifact new
Develop-time artifact reused
Sysnovare Repository of Process Models (Oracle RDB)
Storage
Forward transformation (data flow)
Reverse transformation (data flow)

Figure 6.1: Runtime Dataflow view (artifacts and flow)

1. After invoking the event that starts transformation, the Sysnovare Repository of Process

Models provides a XML file generated by the graphical editor. The XML contains the process model, in Listing 6.1 we can see how a graphical object of "Task" type is stored in the file.

```
<Task name="Atividade do fluxo de trabalho"
    code="A0200"
    enabled="Y"
            table_line_id="3850306750DEV13.01.15 12:14:29,867462 +00:0024216"
    kind="Task"
    type="N"
    eng_id="121"
    gessisync="1"
    description="Atividade do fluxo de trabalho"
    id="3850306750DEV1301151214298674620000024216">
  <mxCell style="shape=ext;rounded=1;strokeColor=#007FFF;whiteSpace=wrap"
              parent="3850306750DEV1301151212112936020000009194"
      vertex="1">
    <mxGeometry x="303" y="62" width="120" height="80" as="geometry"/>
  </mxCell>
</Task>
```

Listing 6.1: Partial XML generated by Graphical Editor

2. Then the XSLT transformation reads the XML, performs the transformation rules defined in a XSLT and writes the model in XMI format in the BPM suite operating system.
3. The model transformation engine reads a XMI file and writes the target model in XMI format, the Sysnovare Model is stored in the OS of BPM suite. In Listing 6.2 we can see that a object element of type "Task" in the Listing 6.1 is mapped to an "Activity", according to the QVT-R rules.

```
<element kind="Activity" xsi:type="WfmMetamodel:Activity"
     xsi:id="3850306750DEV1301151214298674620000024216"
     id="3850306750DEV1301151214298674620000024216"
     type="N"
     code="A0200"
     name="Atividade do fluxo de trabalho"
     eng_id="121"
     mdl_id="3850306750DEV13.01.15 12:12:11,293602 +00:009194"
     refersToModel="3850306750DEV1301151212112936020000009194"
     enabled="Y"
     description="Atividade do fluxo de trabalho"
     ui_left="303" ui_top="62"
     table_line_id="3850306750DEV13.01.15 12:14:29,867462 +00:0024216" />
```

Listing 6.2: Partial XMI generated by Model Tranformation layer

4. The XSLT Transformation reads the XMI file, performs the transformation rules defined in XSLT and writes the new element in XML format of Sysnovare. In Listing 6.3 we can see that the "Activity" element defined earlier in XMI format (see Listing 6.2), changes its data structure. The XML Relational Transformation must recognize the "tablerowdata" as a row of data to be inserted or updated in the table that stores the activities of a process.

```xml
<row>
  <tablerowdata>
      <id>3850306750DEV1301151214298674620000024216</id>
    <type><![CDATA[$N$]]></type>
    <code><![CDATA[$A0200$]]></code>
    <name><![CDATA[$Atividade do fluxo de trabalho$]]></name>
    <eng_id>121</eng_id>
    <mdl_id transformedBy="1305">
      <table_line_id>
        <![CDATA[$3850306750DEV13.01.15 12:12:11,293602 +00:009194$]]>
      </table_line_id>
    </mdl_id>
    <enabled><![CDATA[$Y$]]></enabled>
    <description><![CDATA[$Atividade do fluxo de trabalho$]]></description>
    <ui_left>303</ui_left>
    <ui_top>62</ui_top>
    <table_line_id>
      <![CDATA[$3850306750DEV13.01.15 12:14:29,867462 +00:0024216$]]>
    </table_line_id>
  </tablerowdata>
</row>
```

Listing 6.3: Partial XML generated by XML Transformation layer

5. Finally, the XML that conforms to the structure of the Sysnovare is stored in the suite repositories through the XML Relational Transformation component.

The approach supports the reverse transformation, thus the model transformation is considered bidirectional.

6.2 XML Transformation

As mentioned at the beginning of the chapter, the first layer processes the input and output data before performing the transformation of models. This layer may need less attention in the maintenance and evolution of the models; the objective of this stage is to rearrange the model data and define the information in XMI format.

To implement that layer a set of procedures to get the model and the direction of the transformation were created. Preserving the advantages of the technologies that implement the BPM suite, the layer is developed in PL/SQL in conjunction with XMLType API. Through these technologies the processor accesses the data and structure of the XML document and transforms the model using the rules defined in XSLT. The transformation rules defined in XSLT files are implemented by each identified element.

In Listing 6.4 we can a function in PL/SQL code that executes the XSLT transformations. The function receives a XML file with its filename and the transformation direction as parameters, it executes a first XSL transformation, it invokes the Model Transformation Engine and it executes a second XSL transformation.

```
FUNCTION get_model_transformed ( pxml_finput     IN XMLTYPE,
                                 pv_direction    IN VARCHAR2,
                                 pn_filename     IN NUMBER )
    RETURN XMLTYPE
IS
    lxml_transform_result       XMLTYPE;
    lxml_1st_xsl_transform      XMLTYPE;
    lxml_2nd_xsl_transform      XMLTYPE;
    lxml_input_transform        XMLTYPE;
    lxml_output_transform       XMLTYPE;
    lcl_result                  CLOB;
BEGIN

    /* get first XSL: XML - XMI */
    lxml_1st_xsl_transform  := xmltype( CASE WHEN pv_direction = 'wfm' THEN
        get_xslt_graphic2xmi
                        WHEN pv_direction = 'bpmn' THEN get_xslt_sync2xmi
                        ELSE get_xslt_graphic2xmi END );
    /* XSLT Transformation */
    lxml_input_transform    := pxml_finput.transform ( lxml_1st_xsl_transform );
    /* QVT-R Transformation */
    lcl_result := model_transformation ( pcl_input      => lxml_input_transform.
        getclobval (),
                      pv_direction   => pv_direction,
                      pn_filename    => pn_filename );
                          lxml_transform_result   := xmltype( lcl_result );
    /* get second XSL: XMI - XML */
    lxml_2nd_xsl_transform  :=  xmltype( CASE WHEN pv_direction = 'wfm' THEN
        get_xslt_xmi2sync
                         WHEN pv_direction = 'bpmn' THEN get_xslt_xmi2graphic
                         ELSE get_xslt_xmi2sync END );
    /* XSLT Transformation */
    lxml_output_transform   := lxml_transform_result.transform (
        lxml_2nd_xsl_transform );
    RETURN lxml_output_transform;
    EXCEPTION
      WHEN OTHERS
      THEN
```

```
        gessi_erros_web.regista_erro_oracle;
END get_model_transformed;
```

Listing 6.4: PL/SQL function

To transform the model of the modeling tool to BPM suite model, or towards "bpmn-> wfm", the processor takes the XML file to XMI. After processing and obtaining the target model in XMI format, the processor takes the XMI file to XML. Then, the generated XML is synchronized with the BPM suite database. In the opposite direction, to get the diagram for the model, XML is received from the BPM suite, XMI is converted and after the transformation occurs, one obtains the XML file that the graphical tool draws in the graphical interface. We define this process as transformation towards "wfm-> bpmn".

The rules defined in XSLT language basically define the objects that are part of the metamodel, identify the attributes of each element and represent the associations among the elements belonging to the model. As we can see in Listing 6.5, a rule in XSLT generates a object that represents a graphic element into an element in XMI format.

In Figure 6.2 and Listing 6.1, shown in the Section 6.1, we can see a graphic element of type "Task" and its structure equivalent in XML format in the listing.

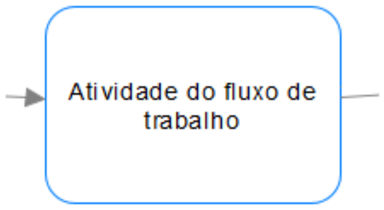


Figure 6.2: Graphical element of "Task" type (in portuguese)

In Listing 6.5, we declaratively define the new format of the element after identifying it, we declare the type of graphic element according to the metamodel which it obeys, we treat cell associations with other elements of the diagram, we treat the geometric properties and finally we copy the data of the specific attributes of BPMN elements.

```
<xsl:for-each select="*">
  <element>
    ...
    <geometry>
      <xsl:for-each select="./mxCell/mxGeometry/@*">
        <xsl:attribute name="{name()}"><xsl:value-of select="." /></xsl:attribute>
      </xsl:for-each>
    </geometry>

    ...

    <!-- Define the value of element :object -->
    <object>
      <xsl:attribute name="xsi:type">BpmnMetamodel:<xsl:value-of select="name(.)" /
          ></xsl:attribute>
      <xsl:attribute name="kind"><xsl:value-of select="name(.)" /></xsl:attribute>
      <xsl:for-each select="@*">
        <xsl:attribute name="{name()}"><xsl:value-of select="." /></xsl:attribute>
      </xsl:for-each>
    </object>
  </element>
</xsl:for-each>
```

Listing 6.5: XSTL Mapping Rule - mxGraph XML2XMI

Finally after performing the transformation rules the result of data are generated in XMI format (see Listing 6.6). The rules for this layer are defined in the Appendix B.

```
<element xsi:id="3850306750DEV13011512142986746200002 4216"
     xsi:type="BpmnMetamodel:Vertex"
     style="shape=ext;rounded=1;strokeColor=#007FFF;whiteSpace=wrap"
     refersToParent="3850306750DEV130115121211293602000091 94"
     parent="3850306750DEV13011512121129360200009194"
     vertex="1">
    <geometry as="geometry" x="303" y="62" width="120" height="80" />
    <object xsi:type="BpmnMetamodel:Task"
          kind="Task"
        name="Atividade do fluxo de trabalho"
        code="A0200"
        enabled="Y"
        table_line_id="3850306750DEV13.01.15 12:14:29,867462 +00:0024216"
        type="N"
        eng_id="121"
        gessisync="1"
        description="Atividade do fluxo de trabalho"
        id="3850306750DEV130115121429867462000024216">
    </object>
</element>
```

Listing 6.6: Partial XMI generated by XSLT Transformation component

6.3 Metamodels

To communicate the first and second layer, the system stores the models in XMI format in the operating file system. To avoid problems in the model editing performed by the transformation engine, the templates are stored in *id_random_model.xmi* format, in which:

- **id**: it is replaced by the unique identifier of the model.

- **random**: it is a random number that serves to differentiate the model if there is more than one application for conversion of the same process.

- **model**: it is a fixed value that depends on whether the model is source or target, depending on the direction of the transformation.

Once the models are available for processing, it is mandatory for the models to conform to their corresponding metamodels. The latter must be in Ecore format to be processed by the transformation engine.

The first metamodel to be identified is the mxGraph BPMN metamodel, to define the structure of data in an Ecore format it was necessary to analyze the structure through the XML created by the modeling tool. Eclipse Modeling Framework Project (EMF) tool allows creating the metamodel in Ecore format through a graphical environment identifying the structure in classes, attributes and associations.

In Figure 6.3 we can see the representation of the mxGraph BPMN metamodel. The structure is mainly composed of two parts: the first part defines the metamodel that mxGraph uses to generate graphs consisting of vertex and edges, and the second defines the BPMN elements. Each element of the graph can be customized through the association of an *DiagramElement* with an *Object*. The *Object* can define the BPMN elements, initially with a set of basic attributes needed to define the behavior in the Sysnovare suite. Finally, the geometric properties of graphical objects are defined in the class *Geometry*.

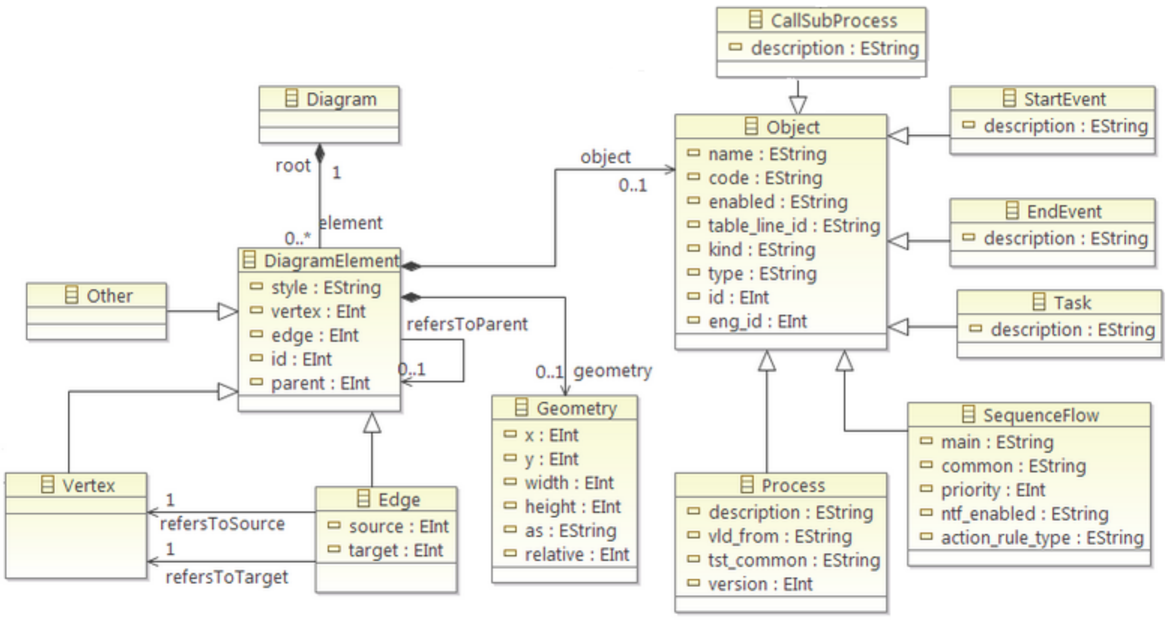


Figure 6.3: mxGraph BPMN Metamodel (Ecore notation)

The Sysnovare Workflow (WF) metamodel will also be necessary to perform model transformations. To set the metamodel is used EMF tool, so that after shaping the structure, the metamodel is obtained in Ecore format. To set the metamodel one must identify the elements defined in the XML of Sysnovare WF.

In Figure 6.4 we can see that the synchronizer groups a set of elements. Since the synchronizer uses a hierarchical structure to indicate the relationships between the objects, in this case the synchronizer can be composed by a *Model*, which in turn is associated to *Activities* and *Transitions*. Each element has a basic set of attributes that will be needed for mapping among elements. In this data structure, each class other than "Synchronization" and "Element" are representations of tables in the database storing the templates.

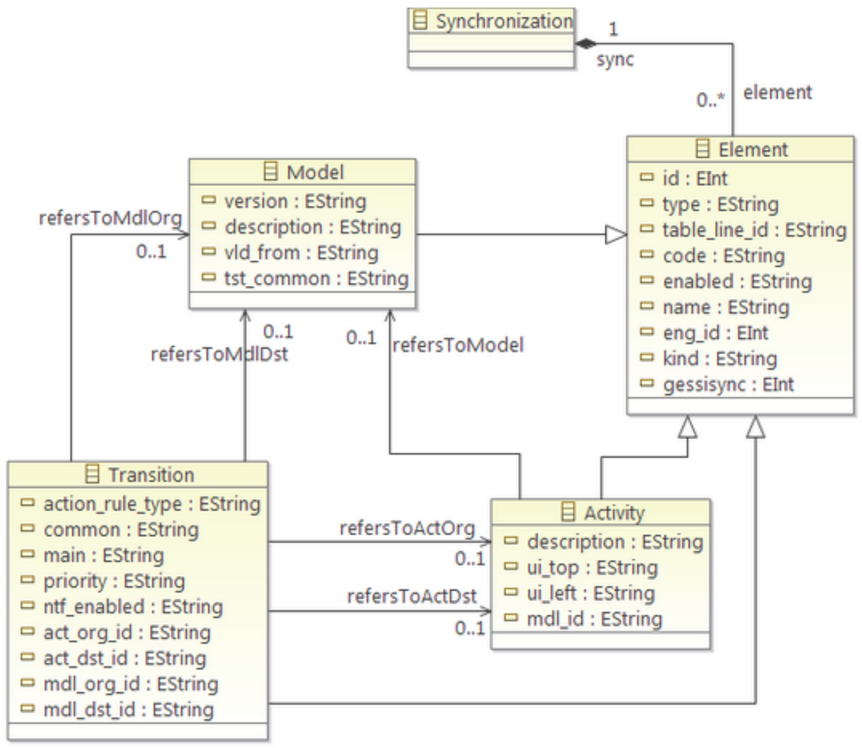


Figure 6.4: Sysnovare Workflow Metamodel (Ecore notation)

Metamodels are incorporated into the created Model Transformation Engine. To transform the metamodel information in code, the EMF tool generates Java code from the Ecore file; it basically creates a set of classes organized into packages.

6.4 Mapping Rules

In a mapping to relate pairs of elements it is necessary to identify the elements and then define the relationship as a rule in the QVT-Relational language. In Figure 6.5 we can see the mapping among the elements of Sysnovare Workflow and BPMN mxGraph elements. The rules are a group of 4 associations, which in turn may be related to other sub rules.

In a QVT-Relational language, a relationship can be performed automatically if the word "top" is used in the rule definition. A "top" relationship is invoked whenever the engine of transformation

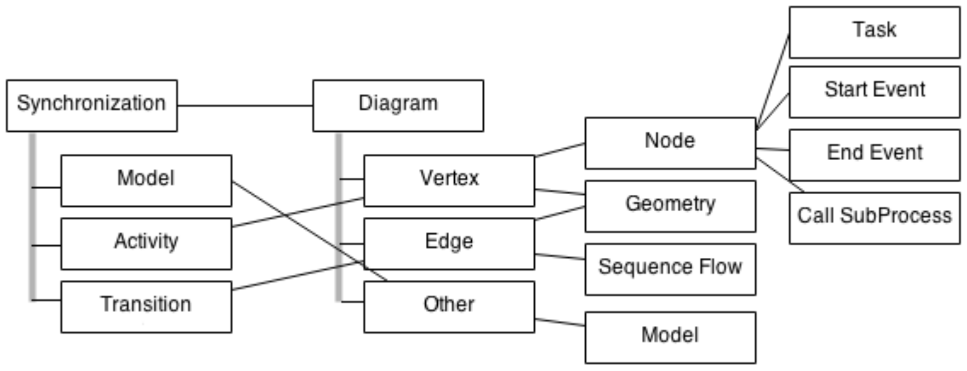


Figure 6.5: Correspondence between Sysnovare Workflow and mxGraph BPMN elements

locates one of the domains of the relationship; this way, the rule is applied to the pair of elements, creating the missing element. On the contrary, if a rule does not possess the reserved word "top", it will only be invoked in case such information is specified in the code.

In each rule you can see the word "enforce"; through this word you specify if the domain is used only for consultation or if it is used to create the object, in the case of not being present in the relationship. When consulting an element, the mode that should be used is "checkonly" to replace the "enforce" mode. Both modes check the consistency of the relationship, so each rule returns "true" if the set of models is consistent according to the transformation and "false" otherwise.

Transformation rules in QVT-R language use the terms "when" and "where"; they define pre-conditions and post-conditions respectively. Each line declared within the clause must be true to run.

In Listing 6.7 we can see an example of a transformation rule. The rule defines that a "package" of UML notation is equivalent to a "schema" in a relational model of a database. The code defines that the transformation is one-way, i.e. each time a package not undefined is identified, a schema element is automatically generated, and this is defined in "enforce" mode. Via "pn" variable, the mapping of the attribute that defines the name of the element is carried out. A birectional transformation is defined if both domains are of type "enforce".

```
top relation PackageToSchema {
    pn : String;
    checkonly domain uml p : SimpleUML::UmlPackage { umlName = pn };
    enforce domain rdbms s : SimpleRDBMS::RdbmsSchema { rdbmsName = pn };
    /* pre-condition*/
    when {  not( p.oclIsUndefined() ) }
}
```

Listing 6.7: Transformation rule for mapping a Package to Schema

The rules defined for this layer are defined in the Appendix C. A rule for each mapping defined in Chapter 4 was created; in some cases, it was necessary to create a rule for each subtype of an element.

6.5 Model Transformation Engine

This sub-component is developed in Java. To set the engine, a Java archive (JAR) from a Java project was created. The engine is intended to run the transformation rules, read the standard input and write the output model. In reading and writing models the engine validates whether the models obey their respective metamodels. For the engine to support transformations expressed in QVT-Relational language, one uses a plug-in available from Medini QVT tool; thus, the engine uses the features of Medini QVT. In Figure 6.6 we can see a logical structure of this component.

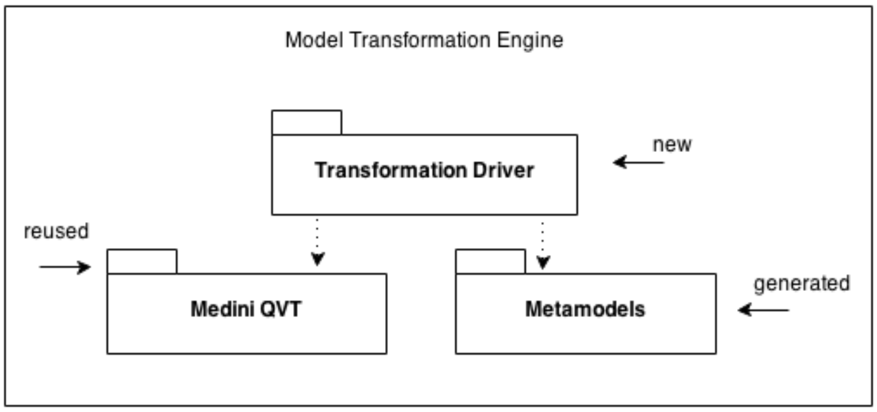


Figure 6.6: Logical Structure of Model Transformation Engine

Listing 6.8 shows tha Java code for the Transformation Driver component. In the function we associate the metamodels and QVT-R rules from file ".qvt". Then, this function reads a input file, it executes the rules on data input invoking the function *evaluateQVT* supported by a plug-in Medini QVT and it creates a output file.

```
/*
 * Based on Java example of the plug-in Medini QVT integration
 */
  public void modelTransformation(String path, String direction, String filename,
      String qvtFilename, String input, String output) {

    // Initialize resource set of models
    this.resourceSet = new ResourceSetImpl();
    this.resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap().put(
        Resource.Factory.Registry.DEFAULT_EXTENSION, new XMIResourceFactoryImpl());

    // Collect all necessary packages from the metamodel(s)
    Collection<EPackage> metaPackages = new ArrayList<EPackage>();
    metaPackages.add(BpmnMetamodelPackage.eINSTANCE);
    metaPackages.add(WfmMetamodelPackage.eINSTANCE);
    // make these packages known to the QVT engine
    this.init(metaPackages);

    // Load the example model files
    Resource resource1 = this.getResource( input );
    Resource resource2 = this.getResource( output );

    // Collect the models, which should participate in the transformation.
    Collection<Collection<Resource>> modelResources = new ArrayList<Collection<
        Resource>>();
    Collection<Resource> firstSetOfModels = new ArrayList<Resource>();
    Collection<Resource> secondSetOfModels = new ArrayList<Resource>();
    modelResources.add(firstSetOfModels);
    modelResources.add(secondSetOfModels);
    firstSetOfModels.add(resource1);
    secondSetOfModels.add(resource2);

    // The directory to store the trace (meta)models
    URI directory = URI.createFileURI( path + "/traces");
    this.preExecution(modelResources, directory);

    // Load the QVT relations
    FileReader qvtRuleSet = null;
    String filename = null;
    try {
      filename = path + "/qvt/" + qvtFilename + ".qvt";
      qvtRuleSet = new FileReader( filename );
    } catch (FileNotFoundException fileNotFoundException) {
      fileNotFoundException.printStackTrace();
      return;
    }
    // Transformation name
    String transformation = qvtFilename;
    // Direction of the transformation
    String dir = direction;
    try {
      // transformation
        this.processorImpl.evaluateQVT(qvtRuleSet, transformation, true, dir, new
            Object[0], null, this.logger);
        this.clean();

    } catch (Throwable throwable) {
      throwable.printStackTrace();
    }

    try {
      resource2.save(Collections.EMPTY_MAP);
    } catch (IOException exception) {
      exception.printStackTrace();
    }
```

```

}
```

Listing 6.8: Java code of Transformation Driver

The implementation of the engine is based on a set of packages with classes that define the metamodel of Sysnovare Workflow and mxGraph BPMN (see Section 6.3). To run the processing engine a procedure in PL/SQL is executed (see Listing 6.9), which executes JAR via the command line. The engine execution requires the following parameters:

1. Direction of transformation: direction "bpmn" if the target model obeys mxGraph BPMN metamodel or "wfm" when the target model obeys Sysnovare Workflow metamodel.

2. Name of the file to be read as input model.

```
PROCEDURE model_transformation ( pv_direction      IN VARCHAR2,
                                  pn_filename       IN NUMBER )
 IS
    lv_filename     VARCHAR2( 100 );
    ln_cmd          NUMBER;
 BEGIN

    SELECT SYSDATE || '_' || random.rand INTO lv_filename FROM v$database;
    lv_filename := pn_filename || '_' || lv_filename;

    ln_cmd := util_caller.run_cmd( wfm_mdl_models_int_tp.gconst_transf_cmd || ' '
         || pv_direction || ' ' || lv_filename  );
 END model_transformation;
```

Listing 6.9: PL/SQL code for execution of Model Transformation layer

In Figure 6.7 we can see that the transforming engine "m2m.jar" is stored in the BPM suite operating system (OS). To facilitate communication between the layers of the Model Transformation and XSLT Transformation, directories are used. The directory "/model" shares models in XMI format and "/qvt" directory stores the bidirectional transformation rules in "bpmn2wfm.qvt" file.

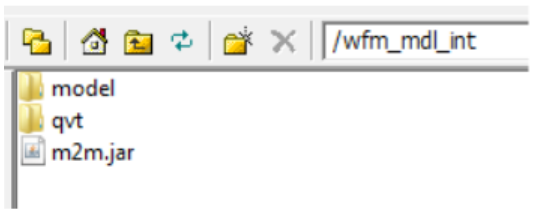


Figure 6.7: Transformation engine integrated in operating system (OS) of BPM suite

6.6 Conclusion

Based on the choices of the different tools which support MDA methodology, it was possible to assure the interoperability between Sysnovare BPM suite and the graphical modeling editor. Throughout this chapter we have described the work done to achieve the external interoperability of BPM suite.

In this chapter we realize that part of the requirements specified in Chapter 4 have been achieved. The final result is an automated system able to communicate in both directions. From a set of previously evaluated technologies and techniques, it was possible to guarantee a system for bidirectional transformation. During implementation, we can see that by means of auxiliary tools it has been possible to facilitate the development of each component. An example case was to use the EMF tool to model metamodels and generate Java code to be integrated in the processing engine.

Implementation was simple in terms of integration, since the technologies of each component are compatible with the existing technologies in BPM suite. The behavior of the transformation tool is presented in a sequence of activities; since this layer is not a visible mechanism to users, it is important to specify how the system behaves.

Finally, we can emphasize that this transformation layer can evolve and be improved. In the continuation of the project, other features can be implemented, as well as adding functions in the system to support data updates in parallel.

Chapter 7

Presentation and Validation from the User Perspective

Two business processes that Sysnovare BPM suite supports will be presented in this chapter. This way, we can assess the usability and performance of the integration of the approach in BPM suite. From this evaluation we check whether there is some improvement in the experience of use in modeling.

Currently, the business processes that Sysnovare suite supports are internal processes and processes for the management of universities like the case of "Universidade de São Paulo". In this chapter we define processes that are modeled, executed and monitored in the suite for testing and validating the approach proposed.

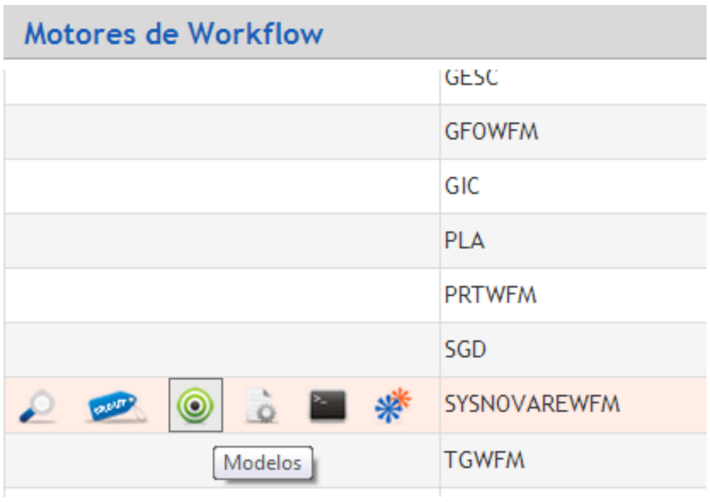
7.1 Human Resources: Holidays Request

Vacation request is a common process in many organizations. This process is simple and it begins when the employee requests his/her vacation period. Then the request is reviewed by the boss or another officer, so that after the analysis there is a notification showing whether the request was accepted or rejected and the process is closed. On the other hand, the employer can cancel the request closing the process.

To model this business process on the BPM suite the user follows these steps:

1. Creating a new model in the BPM suite, begins after choosing an appropriate engine. As the process belongs to an internal process of Sysnovare, the engine chosen is "SYSNOVAREWFM" (see Figure 7.1).

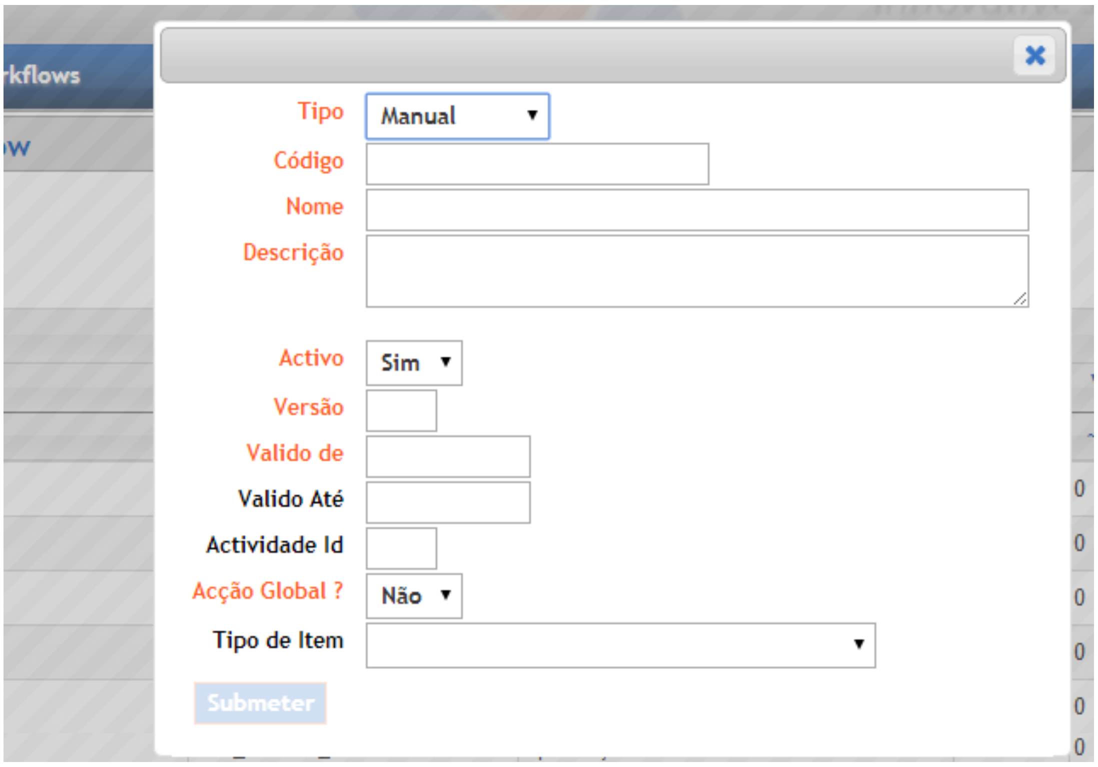
2. Creating a model requires filling in some attributes that define the behavior of the process and other data for informational purposes (see Figure 7.2).



Motores de Workflow

GESC
GFOWFM
GIC
PLA
PRTWFM
SGD
SYSNOVAREWFM
TGWFM

Figure 7.1: Engine selection in BPM suite (in portuguese)

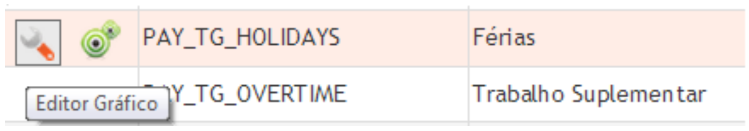


Tipo	Manual
Código	
Nome	
Descrição	
Activo	Sim
Versão	
Valido de	
Valido Até	
Actividade Id	
Acção Global ?	Não
Tipo de Item	

Figure 7.2: Creating the process of vacation request (in portuguese)

3. Editing the model through Diagram Editor modeling tool (see Figure 7.3), allows the user to define the process workflow in a visual environment.

4. After identifying the activities of the business process, identifying the types of elements and defining associations between each element, one gets the workflow that represents the vacation request (see Figure 7.4).



	PAY_TG_HOLIDAYS	Férias
Editor Gráfico	PAY_TG_OVERTIME	Trabalho Suplementar

Figure 7.3: Option to edit a model in Diagram Editor (in portuguese)

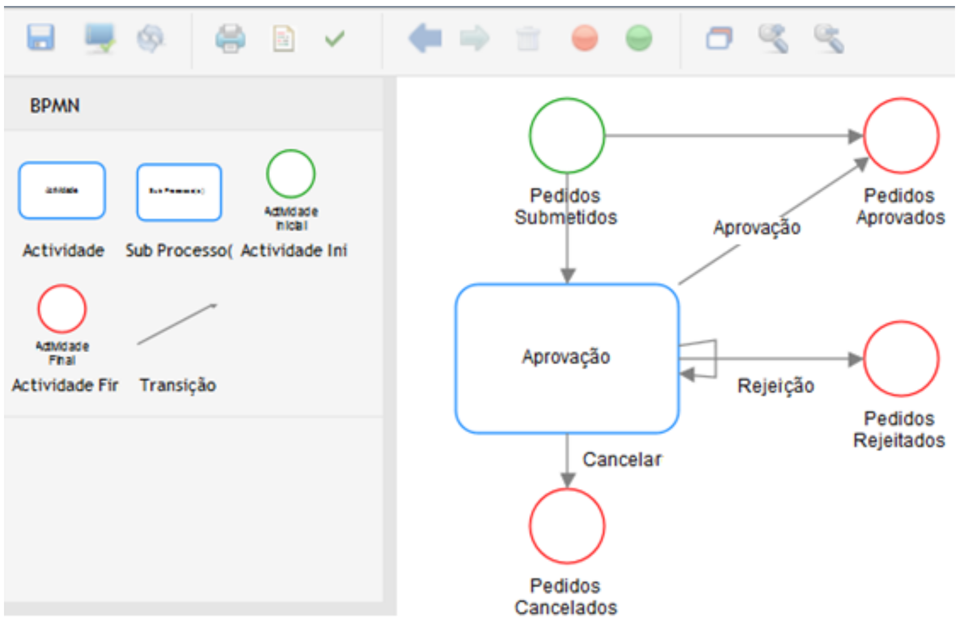
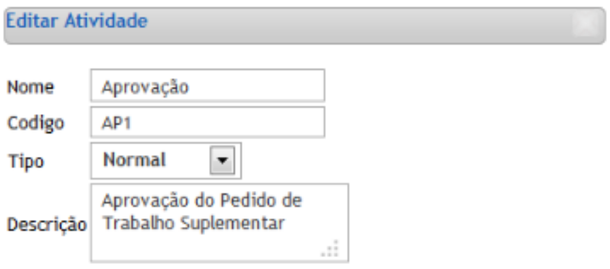


Figure 7.4: Model for vacation request (in portuguese)

Before publishing the model and transforming the BPMN model for mxGraph Sysnovare Workflow model, each graphic element should have a minimum set of attributes filled in. In Figure 7.5 one can see that an activity needs a name, code, type and description. If the attributes are not filled, publication is not possible. Before publishing a model, the tool validates whether the diagram meets the minimum requirements to begin the transformation.



Editar Atividade

Nome	Aprovação
Codigo	AP1
Tipo	Normal
Descrição	Aprovação do Pedido de Trabalho Suplementar

Figure 7.5: Editing basic properties of a graphic element (in portuguese)

5. After performing the transformation, the model data are available in the repositories of the suite. Thus, before running the model it is important to complete the attributes that define the behavior of the activity. Returning to the list of models, the "modeling" option will enable to continue the edition of the properties of the elements of the workflow (see Figure 7.6).

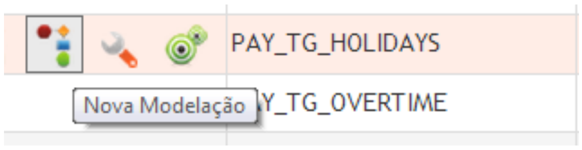
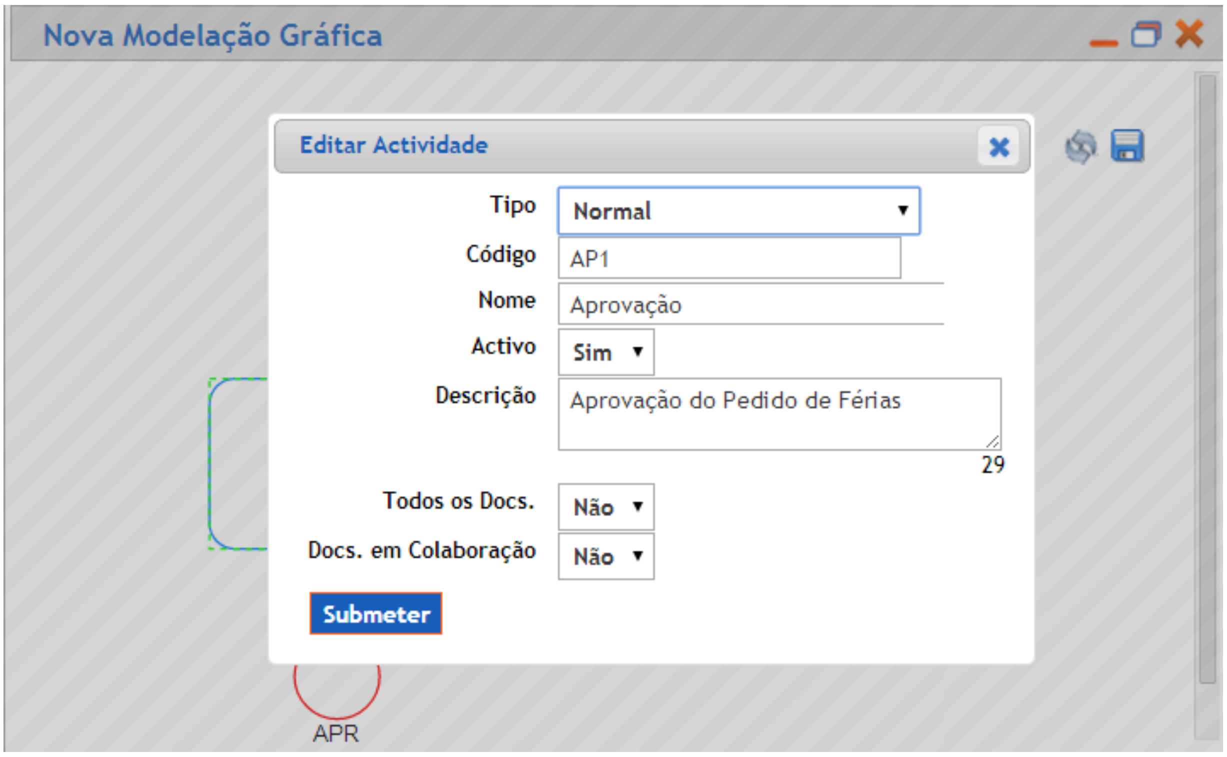


Figure 7.6: Option to work in Property Editor environment (in portuguese)

6. To edit the properties of the workflow, the user must select the element and make the changes. At the end of the process of editing the properties, it is convenient to store the changes in the XML file that stores the diagram (see Figure 7.7). The execution and moni-



Tipo	Normal
Código	AP1
Nome	Aprovação
Activo	Sim
Descrição	Aprovação do Pedido de Férias
	29
Todos os Docs.	Não
Docs. em Colaboração	Não

Figure 7.7: Model editing in the property editor (in portuguese)

toring of the process are performed after modeling.

7. In Figure 7.8 we can see the current state of the workflow via the graph editor in Status Visualizer mode. In this figure the process of holidays request is finished. This process began in *"Aprovação"* task, is executed by a collaborator and it is finished when a boss or HR approve the request. A green color defines when a task or sequence flow was completed.

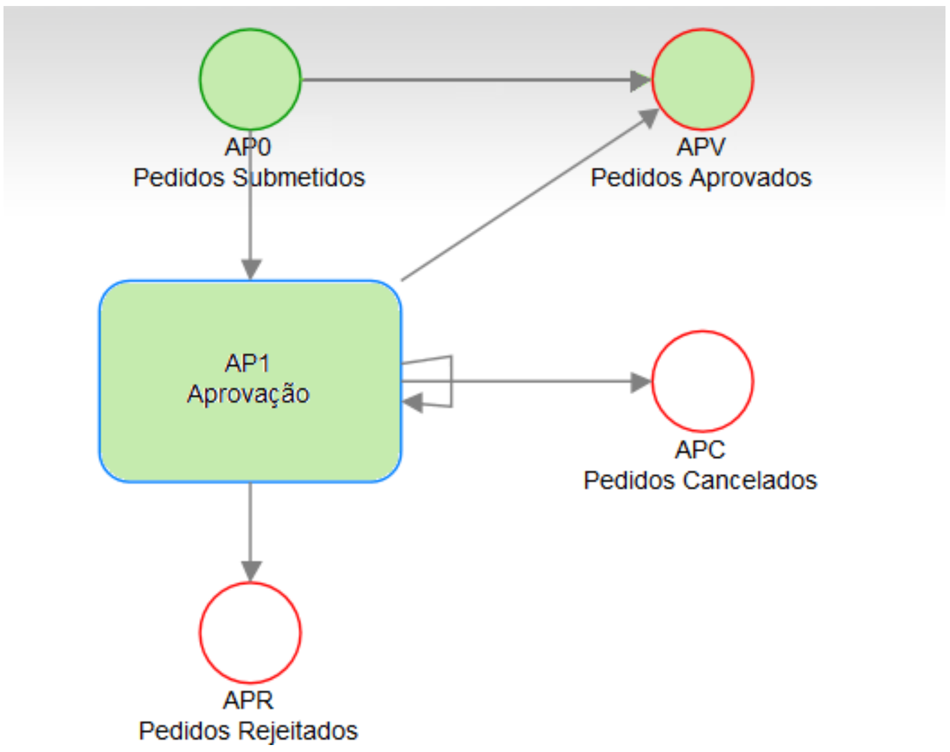


Figure 7.8: Viewing environment of the workflow state (in portuguese)

7.2 Social Action and Education: Equivalence title

The second test case is a process frequently used in universities at the end of each school year; it begins with a phase to accept applications from students who wish to obtain equivalence in their studies. With this process the university recognizes the studies or degree of each student and the entity can categorize whether the student is graduate, master, doctor or the courses that are missing to achieve one of these degrees.

For this test case the transformation of models will be in the opposite direction compared to the first case. Considering that the admission process is created in the BPM suite and it is useful to monitor the entire process, the steps to transform the model are described below:

1. To obtain the diagram of the process it is necessary to open the template through the Graph Editor option. (see Figure 7.3)

2. Then the process diagram is built (see Figure 7.9).

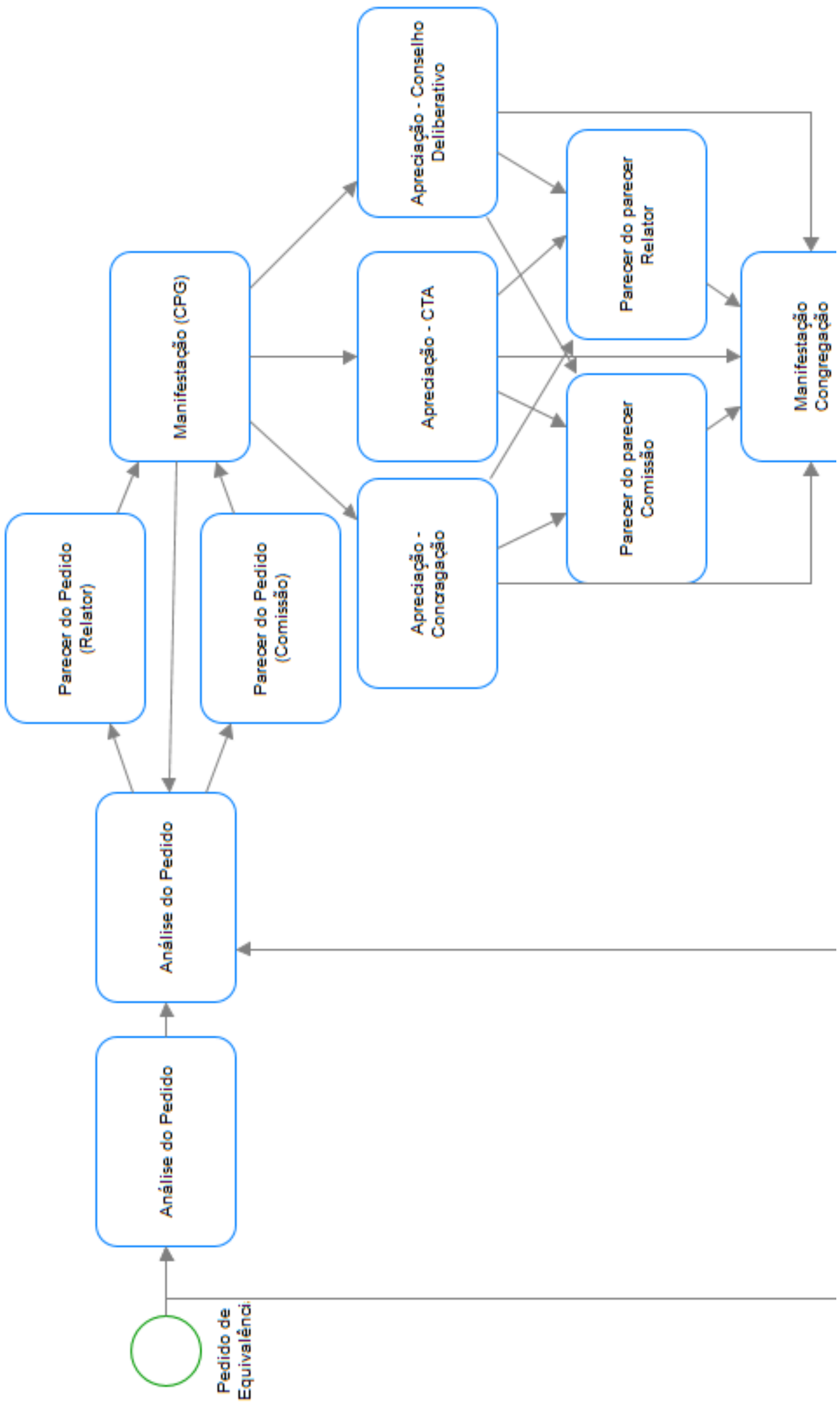

Pedido de Equivalênci
Análise do Pedido
Análise do Pedido
Parecer do Pedido (Relator)
Parecer do Pedido (Comissão)
Manifestação (CPG)
Apreciação - Conselho Deliberativo
Apreciação - CTA
Apreciação - Concragação
Parecer do parecer Relator
Parecer do parecer Comissão
Manifestação Congregação

Figure 7.9: Equivalence title partial model in Diagram Editor (in portuguese)

7.3 Usability and Performance Evaluation

After presenting the modeling experience of two business processes supported by BPM suite, this section presents an overview of the evaluation of the approach in terms of usability and performance.

The usability of the modeling component is improved by integrating a graphical tool that specializes in modeling processes. After some experiments with the new modeling tool, we have identified the following improvements in the experience of using the graphical editor:

- The new modeling component is a complete tool, the BPM suite users can to model a business process in the dinamic graphical environment. The users don't need to use other graphical tools.

- The modeling environment becomes more dynamic from the user's perspective. Through the support of the technique of "drag and drop", you can select a graphic object and drag it onto the diagram. Moving objects and associating them also improves the experience of modeling, since this component had bugs in the previous modeling approach.

- The tool creates a perceptible model diagram. Through the customization of graphics it is possible to characterize their behavior and as a result we obtain a graphical description of the behavior of the workflow.

- It provides extra functionalities with utility in modeling experience as well as printing the design for communication with customers; it permits to move forward or backward in the modeling process, so we can retrieve defined elements; it increases and decreases the view of the diagram, and lastly, it checks whether the diagram is valid to be published later.

Regarding the performance of the integration system, in the execution of a transformation in either direction, transformation time does not exceed the "minute". After modeling both cases described in the previous sections and other processes with different amounts of graphics, the average time was 30 seconds. Since the system is implemented on a web platform, the obtained average is considered a reasonable time interval.

Finally, after presenting the evaluation of the usability and performance of the approach, we can identify an evolution in the modeling component. Usage experience has been improved contributing to a dynamic and intuitive modeling environment.

Chapter 8

Conclusions and Future Work

The presentation of how the project was born, the necessary measures to establish an approach and specification of how the project was developed, summarize the results in this chapter. The reflection on the work done during the project describes how the goals set were achieved. It also describes the future work that can be performed in the project based on the findings obtained.

8.1 Achievements

The project was born through the challenge proposed by Sysnovare - investing in the improvement of one of their latest products, Sysnovare BPM suite. Thus, Sysnovare aims to improve the modeling of a business process, specifically, the experience of using the graphical tool to model business processes. Accepting the challenge and analyzing the current status of the case study, it has been possible to set the basis to contribute positively to the modeling component and partly on the monitoring component of the BPM suite.

Maintaining the competitive advantages of the approach of the BPM suite, by applying the practices of Model-Driven Development (MDD), presents a challenge in the process of defining a proposal that achieves the main goal. Model-Driven Architecture is an interesting approach to the problem context; it is an area with several proposed solutions to establish interoperability between systems. There are few cases study that allow to apply MDA practices for models transformation for runtime interoperability. A bidirectional transformation approach in this case study shows that a viable solution based on transformation of business process models can be used to meet the modeling needs of BPM suite user in real-time. The ability to set a system based on technologies and techniques with considerable maturity to be used in the runtime interoperability increases the level of challenge.

Dividing the main goal into smaller goals has enabled us to define the resources that are better suited to the case study. The communication between the BPM suite and an external modeling tool that replaces the current modeling has been possible through systematization and automation in the interoperability between systems. After the user performs a model transformation, communication effectiveness is visible through the graphical interface. The effectiveness of the transformation process is present in the data consistency and a response delivery to the user within a reasonable time. Achieving this goal has been made possible by an analysis and definition of evaluation criteria to select the best technologies and techniques that have contributed to defining the approach.

During work development we have defined a process of analysis that allows addressing the problem of specifying a mechanism for communication between systems. Defining the requirements and limitations of the case study has enabled us to identify the current resources that the approach can enjoy and set as solution bases. The selection of a basic set of data that will be shared in communication, through the assessment of a transformation metamodel, has allowed a prior evaluation of the approach. Without much effort into mapping elements, the prior evaluation has helped to avoid exploring unnecessary paths. Selecting a tool through an evaluation and presentation of some of them allows the reader to categorize the context in which those tools may be useful. By identifying the competitive advantages of each tool displayed, it is possible to obtain a reference of the applications used in the field of BPM. Finally, defining the most suitable approaches to integrate the solution allows us to present a general evaluation of the technologies more thoroughly investigated in the MDA area. Thus, we have contributed with an overview of the applicability of the approaches.

The main objective of the project has been achieved through a transparent communication system. At the same time presenting a case study to the academic and industrial community, has allowed serving a second purpose. The project also works as a practical example of the applicability of the MDA model transformation approaches. Thus, it contributes with a real case application of MDA methodologies in the area of Business Process Management for runtime interoperability systems.

Finally, we believe that following a simple approach, like adding new features of graphic design and fixing the bugs in the graphical environment of Sysnovare current modeling, is not a viable solution and does not profit from the advantages of MDA. The approach presented allows enjoying the advantages of a graphical tool. Through interoperability between systems, without major development efforts, we have created an intuitive modeling environment and of greater usability to users of BPM suite.

8.2 Future Work

One of the advantages of applying the methodologies of Model-Driven Architecture in software is to develop solutions without major efforts of implementation. Many companies are involved in the investigation of this approach in order to apply the advantages of MDA in their products. For some of those organizations the applicability of these practices is considered as MDA strategy to increase the competitive advantages of its solutions. Sysnovare is part of this group of companies;

through the case study we have verified the necessary investigation to introduce MDA techniques in Sysnovare BPM suite.

Applying the advantages of MDA in Sysnovare BPM suite is an evolving process. The approach presented solves the problem of Sysnovare; however, the approach can be improved so as to enable incremental transformation. For example, the usefulness of this type of transformation can be applied in approaches where there are updates of data in parallel.

The architecture of the approach has been set so that the layers can be adapted in an atomic form. Thus, the way for the transformation component to evolve includes work on the following: Input and Output Processing and Transformation Engine. Depending on the adjustments required, other components may be supplemented or amended in the same way as the two components mentioned above.

An incremental processing can increase the speed with which changes in the diagram of the model are presented to the user. In the Property Editor mode there are no restrictions on the editing of attributes. There are properties that influence the design characteristics of the model, and editing is not restricted, the user is not limited to set new graphic features through this mode. After performing the action "save" in the Property Editor mode, the user performs the transformation in reverse, so you can maintain data consistency. However, the incremental change from this mode can improve the approach, so that the proposal is more complete and sophisticated. After editing each graphic object, the changes of the graphical diagram can be displayed automatically.

The evolution of the solution can be related to the implementation of extra functionalities on the system so that the processing system is able to recognize a change in their behavior. In the Property Editor environment, the transformation tool could automatically distinguish the need to perform an incremental transformation. So, after you edit the properties of each element, the changes made to the diagram of the model will automatically be displayed. In a second phase, to continue the evolution of the project, research could focus on adapting the approach to support incremental transformation.

References

[AHW03] WilM.P. Aalst, ArthurH.M. Hofstede, and Mathias Weske. Business process management: A survey. In WilM.P. Aalst and Mathias Weske, editors, *Business Process Management*, volume 2678 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2003.

[All10] Thomas Allweyer. *BPMN 2.0: Introduction to the Standard for Business Process Modeling*. BoD–Books on Demand, 2010.

[ARC$^+$13] Corina Abdelahad, Daniel Riesco, Alessandro Carrara, Carlo Comin, and Carlos Kavka. Towards the standardization of industrial scientific and engineering workflows with qvt transformations. *International Journal On Advances in Software*, 6(1 and 2):131–141, 2013.

[BCT05] AlanW. Brown, Jim Conallen, and Dave Tropeano. Introduction: Models, modeling, and model-driven architecture (mda). In Sami Beydeda, Matthias Book, and Volker Gruhn, editors, *Model-Driven Software Development*, pages 1–16. Springer Berlin Heidelberg, 2005.

[Biz13] Bizagi. Bizagi Process Modeler. Available at `http://www.bizagi.com/index.php/products/bizagi-process-modeler`, July 2013.

[BRU00] Jörg Becker, Michael Rosemann, and Christoph Uthmann. Guidelines of business process modeling. In Wil Aalst, Jörg Desel, and Andreas Oberweis, editors, *Business Process Management*, volume 1806 of *Lecture Notes in Computer Science*, pages 30–49. Springer Berlin Heidelberg, 2000.

[Fuj13] Fujaba Development Group. Universitat Paderborn | Intallation. Available at `http://www.fujaba.de/resources/installation.html`, November 2013.

[Gar13] Gartner, Inc. Technology Research | Gartner Inc. Available at `http://www.gartner.com/technology/home.jsp`, July 2013.

[Ikv13] Ikv++ technologies ag. Medini QVT. Available at `http://projects.ikv.de/qvt`, November 2013.

[JABK08] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. Atl: A model transformation tool. *Science of Computer Programming*, 72(1–2):31 – 39, 2008. Special Issue on Second issue of experimental software and toolkits (EST).

[JGr13a] JGraph Ltd. Draw.io. Available at `https://www.draw.io`, September 2013.

[JGr13b] JGraph Ltd. JavaScript HTML5 Diagramming Library Component. Available at `http://www.jgraph.com/`, September 2013.

[Ken02] Stuart Kent. Model driven engineering. In Michael Butler, Luigia Petre, and Kaisa Sere, editors, *Integrated Formal Methods*, volume 2335 of *Lecture Notes in Computer Science*, pages 286–298. Springer Berlin Heidelberg, 2002.

[Kur08] I. Kurtev. State of the art of qvt: A model transformation language standard. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5088 LNCS:377–393, 2008. cited By (since 1996)10.

[Lid11] Stephen W Liddle. Model-driven software development. In *Handbook of Conceptual Modeling*, pages 17–54. Springer, 2011.

[MK07] Marion Murzek and Gerhard Kramler. Business process model transformation issues. In *Proceedings of the 9th International Conference on Enterprise Information Systems, Madeira, Portugal*, 2007.

[OMG08] OMG. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification Version 1.0. http://www.omg.org/spec/QVT/1.0/PDF/, April 2008.

[OMG11] OMG. Business Process Model and Notation (BPMN), Version 2.0, January 2011.

[Ora13] Oracle Corporation. XMLType API for PL/SQL. Available at http://docs.oracle.com/cd/B10501_01/appdev.920/a96616/arxml24.htm#1013145, December 2013.

[Paw11] Shefali N. Pawar. Bpmn tools—a comparative analysis to improve interoperability, 2011. Copyright - Copyright ProQuest, UMI Dissertations Publishing 2011; Last updated - 2014-01-13; First page - n/a; M3: M.S.

[SH10] Jim Sinur and Janelle B Hill. Magic quadrant for business process management suites. *Gartner Research, Pub-Date*, 2010.

[Str08] Michael Strommer. Model transformation by-example. May 2008.

[The13a] The Eclipse Foundation. ATL. Available at http://www.eclipse.org/atl/, November 2013.

[The13b] The Eclipse Foundation. Eclipse BPMN2 Modeler. Available at http://www.eclipse.org/bpmn2-modeler/, November 2013.

[The13c] The Eclipse Foundation. Eclipse Modeling - EMF - Home. Available at http://www.eclipse.org/modeling/emf/, November 2013.

[TIB13] TIBCO Software Inc. TIBCO Business Studio | TIBCO. Available at http://www.tibco.com/products/automation/business-process-management/process-modeling/business-studio/default.jsp, November 2013.

[TVZ12] N. Tankovic, D. Vukotic, and M. Zagar. Rethinking model driven development: analysis and opportunities. In *Information Technology Interfaces (ITI), Proceedings of the ITI 2012 34th International Conference on*, pages 505–510, June 2012.

Appendix A

Tools Selection: Summary Tables

Table A.1: Modeling Tool: Summary of evaluation

Criteria / Tool	Popularity / Maturity	Extensibility / Customization	Modeling	Usability
Bizagi Process Modeler	+++ (BPM community)	++ (extension of BPMN elements)	+++ (ex: dynamic association of elements)	+++ (desktop enviroment)
Draw.io	+++	+++ (extension of any element)	+++ (ex: drag-and-drop features)	+++ (web enviroment)
TIBCO Business Studio	+++ (BPM community)	+ (customization of BPMN elements)	+++ (ex: editing attributes)	++ (based on Eclipse project))
BPMN2 Modeler	+++ (Eclipse community)	++ (extension of BPMN elements)	+++ (ex: editing attributes)	++ (based on Eclipse project)

Table A.2: Model Transformation Technology: Summary of evaluation

Criteria / **Approach**	**Popularity / Maturity**	**Integration with other technologies**	**Expressive Power**	**Maintainability**
QVT-R supported by Medini QVT	++ (OMG standard)	+++ (Java plug-in)	+++ (declarative)	+++ (bidirectional)
ATL supported by M2M Eclipse	+++ (older technology)	+++ (Java plug-in)	++ (declarative and imperative)	++ (unidirectional)
XSLT	+++ (older technology)	+++ (API for PL/SQL)	+ (complex syntax)	++ (unidirectional)
TGG supported by Fujaba	++	++ (Java code)	+++ (graphical display)	++ (bidirectional)

Appendix B

XSLT Transformation Rules Code

```
<xsl:template match="/mxGraphModel/root">
  <!-- Header Format -->
  <BpmnMetamodel:Diagram xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:BpmnMetamodel="
      urn:BpmnMetamodel.ecore" xsi:schemaLocation="urn:BpmnMetamodel.ecore ../model
      /BpmnMetamodel.ecore">
  <!-- Search all elements with object -->
  <xsl:for-each select="*">
    <!-- Create a "element" for each diagram element -->
    <xsl:text>&#10;</xsl:text><element>
      <!-- Define all attributes of an diagram element -->
      <xsl:attribute name="xmi:id"><xsl:value-of select="@id" /></xsl:attribute>
      <xsl:choose>
      <xsl:when test="./mxCell/@vertex">
        <xsl:attribute name="xsi:type">BpmnMetamodel:Vertex</xsl:attribute>
      </xsl:when>
      <xsl:when test="./mxCell/@edge">
        <xsl:attribute name="xsi:type">BpmnMetamodel:Edge</xsl:attribute>
      </xsl:when>
      <xsl:otherwise>
        <xsl:attribute name="xsi:type">BpmnMetamodel:None</xsl:attribute>
      </xsl:otherwise>
      </xsl:choose>
          <xsl:when test="name(.) = 'parent'">
          <xsl:attribute name="refersToParent"><xsl:value-of select="." /></
              xsl:attribute>
          <xsl:attribute name="{name()}"><xsl:value-of select="." /></xsl:attribute
              >
        </xsl:when>
        <xsl:when test="name(.) = 'source'">
        <xsl:attribute name="refersToSource"><xsl:value-of select="." /></
            xsl:attribute>
        <xsl:attribute name="{name()}"><xsl:value-of select="." /></xsl:attribute>
        </xsl:when>
        <xsl:when test="name(.) = 'target'">
        <xsl:attribute name="refersToTarget"><xsl:value-of select="." /></
            xsl:attribute>
        <xsl:attribute name="{name()}"><xsl:value-of select="." /></xsl:attribute>
        </xsl:when>
        <xsl:otherwise>
        <xsl:attribute name="{name()}"><xsl:value-of select="." /></xsl:attribute>
        </xsl:otherwise>
      </xsl:choose>
      </xsl:for-each>
      <xsl:if test="not(contains(name(.), 'mxCell'))">
      <!-- Define the properties geometrics of element :geometry -->
      <xsl:if test="./mxCell/mxGeometry">
        <xsl:text>&#10;</xsl:text><geometry>
        <xsl:for-each select="./mxCell/mxGeometry/@*[name()='as' or name()='x' or
            name()='y' or name()='width' or name()='height' or name()='relative']">
          <xsl:attribute name="{name()}"><xsl:value-of select="." /></xsl:attribute
             >
            </xsl:for-each>
        </geometry>
      </xsl:if>
      <!-- Define the value of element :object -->
      <xsl:text>&#10;</xsl:text><object>
        <xsl:attribute name="xsi:type">BpmnMetamodel:<xsl:value-of select="name(.)"
             /></xsl:attribute>
        <xsl:attribute name="kind"><xsl:value-of select="name(.)" /></xsl:attribute
            >
        <xsl:for-each select="@*">
        <xsl:attribute name="{name()}"><xsl:value-of select="." /></xsl:attribute>
        </xsl:for-each>
      </object>
      </xsl:if>
    <!-- Finish of diagram element -->
    </element>
  </xsl:for-each>
  <!-- Footer Format -->
  <xsl:text>&#10;</xsl:text></BpmnMetamodel:Diagram>
</xsl:template>
```

Listing B.1: mxGraph BPMN XML2XMI

```
<xsl:template match="/*[local-name() = 'Diagram']">
  <!-- Header Format -->
  <mxGraphModel grid="1" guides="1" tooltips="1" connect="1" fold="1" page="0"
      pageScale="1" pageWidth="826" pageHeight="1169">
  <root>
  <!-- Define root element of diagram -->
  <mxCell id="0"/>
  <!-- Search all cells of graph-->
  <xsl:for-each select="./element">
    <xsl:choose>
    <!-- Cell with object -->
    <xsl:when test="./object">
      <!-- Get object name -->
      <xsl:variable name="name" select="./object/@kind"/>
      <xsl:element name="{$name}">
      <!-- Get object attributes -->
      <xsl:for-each select="./object/@*">
        <xsl:if test="not(contains(name(), ':')) and not(contains(name(), 'refers')
            )">
        <xsl:attribute name="{name()}"><xsl:value-of select="." /></xsl:attribute>
        </xsl:if>
      </xsl:for-each>
      <!-- Define cell and properties -->
      <mxCell>
        <xsl:if test="$name = 'Process'">
        <xsl:attribute name="parent">0</xsl:attribute>
        </xsl:if>
        <xsl:for-each select="@*">
        <xsl:if test="not(contains(name(), ':')) and not(contains(name(), 'refers')
            )">
          <xsl:attribute name="{name()}"><xsl:value-of select="." /></xsl:attribute
              >
        </xsl:if>
        </xsl:for-each>
        <!-- Define geometry of cell and properties -->
        <xsl:if test="./geometry">
        <mxGeometry>
          <xsl:for-each select="./geometry/@*">
          <xsl:attribute name="{name()}"><xsl:value-of select="." /></xsl:attribute
              >
          </xsl:for-each>
        </mxGeometry>
        </xsl:if>
      </mxCell>
      </xsl:element>
    </xsl:when>
    </xsl:choose>
  </xsl:for-each>
  </root></mxGraphModel>
</xsl:template>
```

Listing B.2: mxGraph BPMN XMI2XML

```
<xsl:template match="/gessisync">
```

```
<!-- Header Format -->
<WfmMetamodel:Synchronization xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:WfmMetamodel="urn:WfmMetamodel.ecore" xsi:schemaLocation="urn:WfmMetamodel.ecore ../model/WfmMetamodel.ecore">
<!-- Get a table data of a model -->
<xsl:for-each select="./wfm_mdl_models/row">
 <xsl:variable name="idMdl" select="translate(translate(./tablerowdata/table_line_id, ’$,.:+’, ’’), ’ ’, ’’)"/>
 <xsl:text>&#10;</xsl:text><element>
 <xsl:attribute name="kind">Model</xsl:attribute>
 <xsl:attribute name="xsi:type">WfmMetamodel:Model</xsl:attribute>
 <xsl:attribute name="xmi:id"><xsl:value-of select="translate(translate(./tablerowdata/table_line_id, ’$,.:+’, ’’), ’ ’, ’’)" /></xsl:attribute>
 <xsl:attribute name="id"><xsl:value-of select="translate(translate(./tablerowdata/table_line_id, ’$,.:+’, ’’), ’ ’, ’’)" /></xsl:attribute>
 <!-- Get a column data of table -->
 <xsl:for-each select="./tablerowdata/*[name()=’name’ or name()=’code’ or name()=’eng_id’ or name()=’enabled’ or name()=’version’ or name()=’vld_from’ or name()=’tst_common’ or name()=’description’ or name()=’type’ or name()=’table_line_id’]">
  <xsl:attribute name="{name()}"><xsl:value-of select="normalize-space(translate(.,’$’,’’))" /></xsl:attribute>
 </xsl:for-each>
 </element>

 <!-- Get a table data of all activities -->
 <xsl:for-each select="./wfm_mdl_activities/row/tablerowdata">
 <xsl:text>&#10;</xsl:text><element>
  <xsl:attribute name="kind">Activity</xsl:attribute>
  <xsl:attribute name="xsi:type">WfmMetamodel:Activity</xsl:attribute>
  <xsl:attribute name="xmi:id"><xsl:value-of select="translate(translate(./table_line_id, ’$,.:+’, ’’), ’ ’, ’’)" /></xsl:attribute>
  <xsl:attribute name="id"><xsl:value-of select="translate(translate(./table_line_id, ’$,.:+’, ’’), ’ ’, ’’)" /></xsl:attribute>

  <!-- Get a column data of table -->
  <xsl:for-each select="./*[name()=’name’ or name()=’code’ or name()=’eng_id’ or name()=’enabled’ or name()=’description’ or name()=’type’ or name()=’table_line_id’ or name()=’mdl_id’ or name()=’ui_left’ or name()=’ui_top’]">
  <xsl:variable name="fkey" select="./table_line_id"/>
  <xsl:choose>
   <xsl:when test="name() = ’mdl_id’ and $fkey">
   <xsl:attribute name="{name()}"><xsl:value-of select="normalize-space(translate($fkey,’$’,’’))" /></xsl:attribute>
   <xsl:attribute name="refersToModel"><xsl:value-of select="translate(translate(../../../../tablerowdata[table_line_id = $fkey]/table_line_id, ’$,.:+’, ’’), ’ ’, ’’)" /></xsl:attribute>
```

```
        </xsl:when>
        <xsl:otherwise>
        <xsl:if test=". != ''">
          <xsl:attribute name="{name()}"><xsl:value-of select="normalize-space(
              translate(.,'$',''))" /></xsl:attribute>
        </xsl:if>
        </xsl:otherwise>
      </xsl:choose>
      </xsl:for-each>
    </element>
    </xsl:for-each>

    <!-- Get a table data of all transitions -->
    <xsl:for-each select="./wfm_mdl_transitions/row/tablerowdata">
    <xsl:text>&#10;</xsl:text><element>
      <xsl:attribute name="kind">Transition</xsl:attribute>
      <xsl:attribute name="xsi:type">WfmMetamodel:Transition</xsl:attribute>
      <xsl:attribute name="xmi:id"><xsl:value-of select="translate(translate(./
          table_line_id, '$,.:+', ''), ' ', '')" /></xsl:attribute>
      <xsl:attribute name="id"><xsl:value-of select="translate(translate(./
          table_line_id, '$,.:+', ''), ' ', '')" /></xsl:attribute>

      <!-- Get a column data of table -->
      <xsl:for-each select="./*[name()='name' or name()='code' or name()='eng_id'
          or name()='enabled' or name()='common' or name()='type' or name()='main'
          or name()='table_line_id' or name()='mdl_org_id' or name()='mdl_dst_id'
          or name()='act_org_id' or name()='act_dst_id' or name()='priority' or
          name()='ntf_enabled' or name()='action_rule_type']">
      <xsl:variable name="fkey" select="./table_line_id"/>
      <xsl:variable name="nKey" select="name(.)"/>

      <xsl:choose>
        <xsl:when test="name() = 'act_org_id' and $fkey">
        <xsl:attribute name="{name()}"><xsl:value-of select="normalize-space(
            translate($fkey,'$',''))" /></xsl:attribute>
        <xsl:attribute name="refersToActOrg"><xsl:value-of select="translate(
            translate(../../../../wfm_mdl_activities/row/tablerowdata[table_line_id
             = $fkey]/table_line_id, '$,.:+', ''), ' ', '')" /></xsl:attribute>
        </xsl:when>
        <xsl:when test="name() = 'act_dst_id' and $fkey">
        <xsl:attribute name="{name()}"><xsl:value-of select="normalize-space(
            translate($fkey,'$',''))" /></xsl:attribute>
        <xsl:attribute name="refersToActDst"><xsl:value-of select="translate(
            translate(../../../../wfm_mdl_activities/row/tablerowdata[table_line_id
             = $fkey]/table_line_id, '$,.:+', ''), ' ', '')" /></xsl:attribute>
        </xsl:when>
        <xsl:when test="name() = 'mdl_org_id' and $fkey">
        <xsl:attribute name="{name()}"><xsl:value-of select="normalize-space(
            translate($fkey,'$',''))" /></xsl:attribute>
        <xsl:attribute name="refersToMdlOrg"><xsl:value-of select="$idMdl" /></
            xsl:attribute>
        </xsl:when>
        <xsl:when test="name() = 'mdl_dst_id' and $fkey">
        <xsl:attribute name="{name()}"><xsl:value-of select="normalize-space(
            translate($fkey,'$',''))" /></xsl:attribute>
        <xsl:attribute name="refersToMdlDst"><xsl:value-of select="$idMdl" /></
            xsl:attribute>
        </xsl:when>
        <xsl:otherwise>
        <xsl:attribute name="{name()}"><xsl:value-of select="normalize-space(
            translate(.,'$',''))" /></xsl:attribute>
        </xsl:otherwise>
      </xsl:choose>

      </xsl:for-each>
    </element>
    </xsl:for-each>
  </xsl:for-each>
  <!-- Footer Format -->
  <xsl:text>&#10;</xsl:text></WfmMetamodel:Synchronization>
</xsl:template>
```

Listing B.3: Sysnovare Workflow XMI2XML

```
<xsl:template match="/*[local-name() = 'Synchronization']">
  <!-- Header Format -->
  <gessisync version="1" struct_id="10">
    <!-- Get a model -->
    <xsl:text>&#10;</xsl:text><wfm_mdl_models>
    <xsl:for-each select="./element[@kind='Model']">
      <xsl:variable name="id" select="@table_line_id"/>
      <!-- Get a model data -->
      <xsl:text>&#10;</xsl:text><row><tablerowdata>
      <!-- Get object attributes -->
      <xsl:for-each select="./@*[not(contains(name(), ':')) and not(contains(name()
          , 'refersTo')) and name() != 'kind' ]">
      <xsl:text>&#10;</xsl:text>
      <xsl:element name="{name()}">
        <xsl:choose>
        <xsl:when test="name() = 'version' or name() = 'eng_id' or name() = 'id' or
             name()='vld_from'">
          <xsl:value-of select="." />
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="concat('&lt;![CDATA[$', concat(.,'$]]&gt;'))" />
        </xsl:otherwise>
        </xsl:choose>
      </xsl:element>
      </xsl:for-each>
      <xsl:text>&#10;</xsl:text></tablerowdata>

      <!-- Get all activities of model -->
      <xsl:text>&#10;</xsl:text><wfm_mdl_activities>
        <xsl:for-each select="../element[@kind ='Activity' and @mdl_id = $id]">
        <xsl:text>&#10;</xsl:text><row><tablerowdata>

          <!-- Get a activity data -->
          <xsl:for-each select="./@*[not(contains(name(), ':')) and not(contains(
              name(), 'refersTo')) and name() != 'kind' ]">
          <xsl:text>&#10;</xsl:text>
          <xsl:element name="{name()}">
            <xsl:choose>
            <xsl:when test="name() = 'mdl_id'">
              <xsl:attribute name="transformedBy">1305</xsl:attribute>
              <table_line_id>
              <xsl:value-of select="concat('&lt;![CDATA[$', concat(.,'$]]&gt;'))" /
                  >
              </table_line_id>
            </xsl:when>
            <xsl:otherwise>
              <xsl:choose>
              <xsl:when test="name() = 'eng_id' or name() = 'id' or name() = '
                  ui_top' or name() = 'ui_left'">
                <xsl:value-of select="." />
              </xsl:when>
              <xsl:otherwise>
                <xsl:value-of select="concat('&lt;![CDATA[$', concat(.,'$]]&gt;'))"
                    />
              </xsl:otherwise>
              </xsl:choose>
            </xsl:otherwise>
            </xsl:choose>
          </xsl:element>
          </xsl:for-each>
        <xsl:text>&#10;</xsl:text></tablerowdata></row>
        </xsl:for-each>
      <xsl:text>&#10;</xsl:text></wfm_mdl_activities>

      <!-- Get all transitions of model -->
      <xsl:text>&#10;</xsl:text><wfm_mdl_transitions>
        <xsl:for-each select="../element[@kind = 'Transition' and @mdl_org_id = $id
             and @mdl_dst_id = $id]">
        <xsl:text>&#10;</xsl:text><row><tablerowdata>

          <!-- Get a transition data -->
          <xsl:for-each select="./@*[not(contains(name(), ':')) and not(contains(
              name(), 'refersTo')) and name() != 'kind' ]">
          <xsl:text>&#10;</xsl:text>
          <xsl:element name="{name()}">
            <xsl:choose>
            <xsl:when test="name() = 'mdl_org_id'">
              <xsl:attribute name="transformedBy">1343</xsl:attribute>
              <table_line_id>
              <xsl:value-of select="concat('&lt;![CDATA[$', concat(.,'$]]&gt;'))" /
                  >
              </table_line_id>
            </xsl:when>
            <xsl:when test="name() = 'mdl_dst_id'">
              <xsl:attribute name="transformedBy">1341</xsl:attribute>
              <table_line_id>
              <xsl:value-of select="concat('&lt;![CDATA[$', concat(.,'$]]&gt;'))" /
                  >
              </table_line_id>
            </xsl:when>
            <xsl:when test="name() = 'act_org_id'">
              <xsl:attribute name="transformedBy">1344</xsl:attribute>
              <table_line_id>
              <xsl:value-of select="concat('&lt;![CDATA[$', concat(.,'$]]&gt;'))" /
                  >
              </table_line_id>
            </xsl:when>
            <xsl:when test="name() = 'act_dst_id'">
              <xsl:attribute name="transformedBy">1342</xsl:attribute>
              <table_line_id>
              <xsl:value-of select="concat('&lt;![CDATA[$', concat(.,'$]]&gt;'))" /
                  >
              </table_line_id>
            </xsl:when>
            <xsl:otherwise>
              <xsl:choose>
              <xsl:when test="name() = 'priority' or name() = 'eng_id' or name() =
                  'id'">
                <xsl:value-of select="." />
              </xsl:when>
              <xsl:otherwise>
                <xsl:value-of select="concat('&lt;![CDATA[$', concat(.,'$]]&gt;'))"
                    />
              </xsl:otherwise>
              </xsl:choose>
            </xsl:otherwise>
            </xsl:choose>
          </xsl:element>
          </xsl:for-each>
        <xsl:text>&#10;</xsl:text></tablerowdata></row>
        </xsl:for-each>
      <xsl:text>&#10;</xsl:text></wfm_mdl_transitions>
      <xsl:text>&#10;</xsl:text></row>
    </xsl:for-each>
    <xsl:text>&#10;</xsl:text></wfm_mdl_models>
  <!-- Footer Format -->
  <xsl:text>&#10;</xsl:text></gessisync>
</xsl:template>
```

Listing B.4: Sysnovare Workflow XML2XMI

XSLT Transformation Rules Code

Appendix C

QVT - Relational Rules Code

```
top relation SynchronizationToDiagram {
     enforce domain wfm s : WfmMetamodel::Synchronization{};
     enforce domain bpmn d : BpmnMetamodel::Diagram{};
}
```

Listing C.1: Map each persistent Data synchronization To Diagram

```
  top relation ModelToProcess {

    c1, c2, c3, c4, c6, c7, c8, c9, c10, c11: String;
    c5, c12: Integer;

      enforce domain wfm m : WfmMetamodel::Model {
        sync = s : WfmMetamodel::Synchronization {},
        kind = 'Model',
        id = c1,
      enabled = c2,
      code = c3,
      name = c4,
      eng_id = c5,
      description = c6,
      version = c7,
      tst_common = c8,
      vld_from = c9,
      type = c10,
      table_line_id = c11,
      gessisync = c12
    };

    enforce domain bpmn n : BpmnMetamodel::None {
        root = d : BpmnMetamodel::Diagram {},
      object = o : BpmnMetamodel::Process {
        kind = 'Process',
        id = c1,
        enabled = c2,
        code = c3,
        name = c4,
        eng_id = c5,
        description = c6,
        version = c7,
        tst_common = c8,
        vld_from = c9,
        type = c10,
        table_line_id = c11,
        gessisync = c12
      }
    };

    when {
        SynchronizationToDiagram( s, d );
    }
  }
```

Listing C.2: Map each persistent Model To Process

```
  top relation ActivityToVertex {

     checkonly domain wfm e : WfmMetamodel::Activity {};

     checkonly domain bpmn de : BpmnMetamodel::Vertex {};

     where {
         ActivityToTask ( e, de ) or
         ActivityToStartEvent ( e, de ) or
         ActivityToEndEvent ( e, de ) or
         ActivityToSubprocess ( e, de );
     }

  }

  }
```

Listing C.3: Map each persistent Activity To Vertex

```
  relation ActivityToTask {

    c1, c2, c3, c4, c6, c8, c9, c10, c11, c12, c13, c14: String;
    c0, c5, c7, c15, c16: Integer;
```

```

      enforce domain wfm a : WfmMetamodel::Activity {
          sync = s : WfmMetamodel::Synchronization {},
        refersToModel = m : WfmMetamodel::Model {},
          kind = 'Activity',
          id = c1,
      enabled = c2,
      code = c3,
      name = c4,
      eng_id = c5,
      description = c6,
      gessisync = c7,
      type = c8,
      table_line_id = c9,
      ui_top = c10,
      ui_left = c11,
      mdl_id = c12
    };

    enforce domain bpmn v : BpmnMetamodel::Vertex {
      vertex = 1,
      visible = c0,
        root = d : BpmnMetamodel::Diagram {},
      refersToParent = p : BpmnMetamodel::None {},
      geometry = g : BpmnMetamodel::Geometry {
        x = c11,
        y = c10,
        width = c15,
        height = c16,
        as = 'geometry'
      },
        object = o : BpmnMetamodel::Task {
        kind = 'Task',
          id = c1,
        enabled = c2,
        code = c3,
        name = c4,
        eng_id = c5,
        description = c6,
        gessisync = c7,
        type = c8,
        table_line_id = c9
        },
        parent = c13,
        style = c14
    };

    when {
        ( not( a.oclIsUndefined() ) and a.type = 'N' )
```

```
          or  not( o.oclIsUndefined());
        SynchronizationToDiagram( s, d );
        ModelToProcess( m, p );
    }

    where {
        p.object.table_line_id = c12;
        m.id = c13;
        if c2 = 'Y'
          then 1
        else 0 endif
          = c0;
        if not ( v.style.oclIsUndefined() ) and v.style.size() > 0
          then v.style
        else 'shape=ext;rounded=1;strokeColor=#007FFF' endif
          = c14;
        if not ( v.geometry.width.oclIsUndefined() ) and v.geometry.width <> 0
          then v.geometry.width
        else 120 endif
          = c15;
        if not ( v.geometry.height.oclIsUndefined() ) and v.geometry.height <> 0
          then v.geometry.height
        else 80 endif
          = c16;
    }
  }
```

Listing C.4: Map each persistent Activity To Task

```
  relation ActivityToStartEvent {

    c1, c2, c3, c4, c6, c7, c8, c9, c10, c11, c12, c13: String;
    c0, c5, c14, c15, c16: Integer;

      enforce domain wfm a : WfmMetamodel::Activity {
          sync = s : WfmMetamodel::Synchronization {},
        refersToModel = m : WfmMetamodel::Model {},
          kind = 'Activity',
          id = c1,
      enabled = c2,
      code = c3,
      name = c4,
      eng_id = c5,
      description = c6,
      type = c8,
      table_line_id = c9,
      ui_top = c10,
      ui_left = c11,
      mdl_id = c12,
      gessisync = c14
    };

    enforce domain bpmn v : BpmnMetamodel::Vertex {
      vertex = 1,
        visible = c0,
        root = d : BpmnMetamodel::Diagram {},
      refersToParent = p : BpmnMetamodel::None {},
      geometry = g : BpmnMetamodel::Geometry {
        x = c11,
        y = c10,
        width = c15,
        height = c16,
        as = 'geometry'
      },
        object = o : BpmnMetamodel::StartEvent {
        kind = 'StartEvent',
          id = c1,
        enabled = c2,
        code = c3,
        name = c4,
        eng_id = c5,
        description = c6,
        type = c8,
        table_line_id = c9,
        gessisync = c14
        },
        style = c7,
        parent = c13
    };

   when {
        ( not( a.oclIsUndefined() ) and a.type = 'I' )
           or not( o.oclIsUndefined() );
        SynchronizationToDiagram( s, d );
        ModelToProcess( m, p );
    }

    where {
        p.object.table_line_id = c12;
        m.id = c13;
        if c2 = 'Y'
          then 1
        else 0 endif
          = c0;
        if not ( v.style.oclIsUndefined() ) and v.style.size() > 0
          then v.style
        else 'ellipse;verticalLabelPosition=bottom;verticalAlign=top;perimeter=
            ellipsePerimeter;outline=standard;symbol=general;strokeColor=#009900'
            endif
          = c7;
        if not ( v.geometry.width.oclIsUndefined() ) and v.geometry.width <> 0
          then v.geometry.width
        else 40 endif
          = c15;
        if not ( v.geometry.height.oclIsUndefined() ) and v.geometry.height <> 0
          then v.geometry.height
        else 40 endif
          = c16;
    }
  }
```

Listing C.5: Map each persistent Activity To StartEvent

```
relation ActivityToEndEvent {

  c1, c2, c3, c4, c6, c7, c8, c9, c10, c11, c12, c13: String;
  c0, c5, c14, c15, c16: Integer;

    enforce domain wfm a : WfmMetamodel::Activity {
        sync = s : WfmMetamodel::Synchronization {},
      refersToModel = m : WfmMetamodel::Model {},
        kind = 'Activity',
        id = c1,
    enabled = c2,
    code = c3,
    name = c4,
    eng_id = c5,
    description = c6,
    type = c8,
    table_line_id = c9,
    ui_top = c10,
    ui_left = c11,
    mdl_id = c12,
    gessisync = c14
  };

  enforce domain bpmn v : BpmnMetamodel::Vertex {
    vertex = 1,
      visible = c0,
      root = d : BpmnMetamodel::Diagram {},
    refersToParent = p : BpmnMetamodel::None {},
    geometry = g : BpmnMetamodel::Geometry {
      x = c11,
      y = c10,
      width = c15,
      height = c16,
      as = 'geometry'
    },
      object = o : BpmnMetamodel::EndEvent {
      kind = 'EndEvent',
        id = c1,
      enabled = c2,
      code = c3,
      name = c4,
      eng_id = c5,
      description = c6,
      type = c8,
      table_line_id = c9,
      gessisync = c14
      },
      style = c7,
    parent = c13
  };

  when {
      ( not( a.oclIsUndefined() ) and a.type = 'F' )
        or  not( o.oclIsUndefined() );
      SynchronizationToDiagram( s, d );
      ModelToProcess( m, p );
  }

  where {
      p.object.table_line_id = c12;
      m.id = c13;
      if c2 = 'Y'
        then 1
      else 0 endif
        = c0;
      if not  ( v.style.oclIsUndefined() ) and v.style.size() > 0
        then v.style
      else 'ellipse;verticalLabelPosition=bottom;verticalAlign=top;perimeter=
          ellipsePerimeter;outline=end;symbol=general;strokeColor=#FF0000' endif
        = c7;
      if not  ( v.geometry.width.oclIsUndefined() ) and v.geometry.width <> 0
        then v.geometry.width
      else 40 endif
        = c15;
      if not  ( v.geometry.height.oclIsUndefined() ) and v.geometry.height <> 0
        then v.geometry.height
      else 40 endif
        = c16;
  }
}
```

Listing C.6: Map each persistent Activity To EndEvent

```
relation ActivityToSubprocess {

  c1, c2, c3, c4, c6, c7, c8, c9, c10, c11, c12, c13: String;
  c0, c5, c14, c15, c16: Integer;

    enforce domain wfm a : WfmMetamodel::Activity {
        sync = s : WfmMetamodel::Synchronization {},
      refersToModel = m : WfmMetamodel::Model {},
        kind = 'Activity',
        id = c1,
    enabled = c2,
    code = c3,
    name = c4,
    eng_id = c5,
    description = c6,
    type = c8,
    table_line_id = c9,
    ui_top = c10,
    ui_left = c11,
    mdl_id = c12,
    gessisync = c14
  };

  enforce domain bpmn v : BpmnMetamodel::Vertex {
    vertex = 1,
    visible = c0,
      root = d : BpmnMetamodel::Diagram {},
    refersToParent = p : BpmnMetamodel::None {},
    geometry = g : BpmnMetamodel::Geometry {
      x = c11,
      y = c10,
        width = c15,
      height = c16,
      as = 'geometry'
    },
      object = o : BpmnMetamodel::CallSubProcess {
      kind = 'CallSubProcess',
        id = c1,
      enabled = c2,
      code = c3,
      name = c4,
      eng_id = c5,
      description = c6,
      type = c8,
      table_line_id = c9,
      gessisync = c14
      },
      style = c7,
      parent = c13
  };

  when {
      ( not( a.oclIsUndefined() ) and ( a.type = 'SWP' or a.type = 'WSWP' ))
        or  not( o.oclIsUndefined() );
      SynchronizationToDiagram( s, d );
      ModelToProcess( m, p );
  }

  where {
      p.object.table_line_id = c12;
      m.id = c13;
      if c2 = 'Y'
      then 1
      else 0 endif
        = c0;
      if not ( v.style.oclIsUndefined() ) and v.style.size() > 0
        then v.style
      else 'shape=ext;rounded=1;strokeWidth=3;strokeColor=#007FFF' endif
        = c7;
      if not ( v.geometry.width.oclIsUndefined() ) and v.geometry.width <> 0
        then v.geometry.width
      else 120 endif
        = c15;
      if not ( v.geometry.height.oclIsUndefined() ) and v.geometry.height <> 0
        then v.geometry.height
      else 80 endif
        = c16;
  }
}
```

Listing C.7: Map each persistent Activity To Subprocess

```
  top relation TransitionToEdge {

    c1, c2, c3, c4, c7, c8, c9, c10, c11, c12, c13, c14, c15, c16, c17, c18, c19,
        c20: String;
    c0, c5, c6 : Integer;

      enforce domain wfm t : WfmMetamodel::Transition {
          sync = s : WfmMetamodel::Synchronization {},
        refersToActOrg = ao : WfmMetamodel::Activity {},
        refersToActDst = ad : WfmMetamodel::Activity {},
        refersToMdlDst = ao.refersToModel,
        refersToMdlOrg = ad.refersToModel,
          kind = 'Transition',
          id = c1,
      enabled = c2,
      code = c3,
      name = c4,
      eng_id = c5,
      gessisync = c6,
      type = c8,
      table_line_id = c9,
      main = c10,
      common = c11,
      ntf_enabled = c12,
      action_rule_type = c13,
      priority = c14,
      mdl_org_id = c16,
      mdl_dst_id = c16,
      act_org_id = c18,
      act_dst_id = c20
    };

    enforce domain bpmn e : BpmnMetamodel::Edge {
      edge = 1,
      visible = c0,
      style = c7,
      parent = c15,
        source = c17,
        target = c19,
        root = d : BpmnMetamodel::Diagram {},
      refersToParent = p : BpmnMetamodel::None {},
      refersToSource = so : BpmnMetamodel::Vertex {},
      refersToTarget = ta : BpmnMetamodel::Vertex {},
      geometry = g : BpmnMetamodel::Geometry {
        relative = 1,
        as = 'geometry'
      },
        object = o : BpmnMetamodel::SequenceFlow {
        kind = 'SequenceFlow',
          id = c1,
        enabled = c2,
        code = c3,
        name = c4,
        eng_id = c5,
        gessisync = c6,
        type = c8,
        table_line_id = c9,
        main = c10,
        common = c11,
        ntf_enabled = c12,
        action_rule_type = c13,
        priority = c14
        }
    };

    when {
         SynchronizationToDiagram( s, d );
         ModelToProcess( ao.refersToModel, p );
         ModelToProcess( ad.refersToModel, p );
         ActivityToVertex( ao, so );
         ActivityToVertex( ad, ta );
    }

    where {
        p.object.table_line_id = c16;
        so.object.table_line_id = c18;
        ta.object.table_line_id = c20;
        ao.refersToModel.id = c15;
        ao.id = c17;
        ad.id = c19;
        if c2 = 'Y'
        then 1
        else 0 endif
          = c0;
        if not ( e.style.oclIsUndefined() ) and e.style.size() > 0
        then e.style
        else 'endArrow=block;endFill=1;endSize=6;verticalAlign=top' endif
          = c7;
    }
  }
```

Listing C.8: Map each persistent Transition To Edge