

# **BUSINESS PROCESS MODELLING AND THE MDA PARADIGM IN PROCESS AUTOMATION AS A COMPETITIVE ADVANTAGE FOR MODERN CORPORATIONS**

Oskeol Gjoni

Delivered  
European University of Tirana  
Doctoral School

Fulfilling the obligations of the Phd program in Economics and Information  
Technology, profile Management, to obtain the scientific degree “Doctor”

Mentor : Prof. Asoc. Dr Indrit Baholli

Number of words: 48400

Tirane, November 2015

## **ABSTRACT**

*Do investments performed from a company in Information Technology projects pay off? Now we have the answer. Yes, if solid business process are in place within the company. Companies no longer compete on the desing of their products but also in the desing of their processes. Solid and managed process provide a competitive advantage for the company. In order to have solid business process a modelling standard is needed that is simple and easily to be understood from all stakeholders (from business to technical people). Also a framework to support modelling, automation and optimization is needed.*

*This work is focussed on the business process management practices within the telecommunications industry in Albania. After analysing the current state of the art of business process management practices in the industry we propose what would be an appropriate framework that would be suitable for the industry. An existing business process within one of the companies is taken into consideration, the user access management process. This process is modelled and automated in two different frameworks Bizagi BPM Suite and MOSKitt. The model driven architecture approach based on models is followed in process automation. Through a comparative analysis are presented the pros and cons of each framework.*

*In the current state of business process management practices in the telecommunication industry in Albania the importance of business process management is recongized from the companies but there are no modelling formalisms and no frameworks used that would support the business analysts in process automation and optimization. We suggest that Bizagi BPM Suite would be a suitable solution for these companies in order to fully embrace the business process management journey.*

*A janë të vlefshme investimet që kryen një kompani në teknologji informacioni dhe a e kthejnë vlerën këto investime për kompaninë? Tani ne kemi një përgjigje për këtë pyetje. Po, këto investime janë të vlefshme nëse kompania ka procese të mirëpërcaktuara dhe të qarta për biznesin. Kompanitë nuk konkurrojnë vetëm në dizeninimin e produktit final por dhe në dizenjimin e proceseve të tyre të biznesit. Procese të qarta dhe të mirëpërcaktuara janë një avantazh konkurrues për kompaninë. Për të pasur procese të mirëpërcaktuara dhe të qarta nevojitet një standard modeli që të jetë i thjeshtë dhe i lehtë për tu kuptuar nga të gjithë aktorët brenda kompanisë (nga njerëzit e biznesit deri tek stafi teknik). Gjithashtu duhet një platformë që të mbështesë modelimin, automatizimin dhe optimizimin e proceseve të biznesit.*

*Ky punim fokusohet në praktikën e menaxhimit të proceseve të biznesit në industrinë e telekomunikacionit në Shqipëri. Pasi analizohet gjënda aktuale a praktikave të menaxhimit të proceseve të biznesit në industri propozohet cila do ishte një platformë e përshtatshme për menaxhimin e proceseve të biznesit për këtë industri. Një proces egzistues biznesi në një nga kompanitë merret në studim, procesi i menaxhimit të aksesit të përdoruesve. Procesi në fjalë modelohet dhe automatizohet në dy platforma të ndryshme Bizagi BPM suite dhe MOSKitt. Paradigma model driven architecture e bazuar mbi modelet është aplikuar për automatizimin e proceseve. Në bazë të një studimi krahasues prezantohen avantazhet dhe disavantazhet e secilës platformë.*

*Në gjëndjen aktuale të praktikave të menaxhimit të proceseve të biznesit në industrinë e telekomunikacionit në Shqipëri rëndësia e menaxhimit të proceseve të biznesit është e pranishme në kompanitë në këtë industri por nuk kemi formalizma modeli dhe platforma që suportojnë analistët e biznesit në automatizimin e proceseve të biznesit dhe optimizimin e tyre. Ne sugjerojmë që Bizagi BPM suite është një platformë e përshtatshme për këto kompani në mënyrë që të aplikojnë praktikën më të mirë në udhëtimin e vazhdueshëm të menaxhimit të proceseve të biznesit.*

## **DEDICATIONS**

*To all the people i love in my life and the ones that have supported me throughout this journey.*





## **TABLE OF CONTENTS**

<b>CHAPTER I: INTRODUCTION .....</b>	<b>13</b>
1.1 Business Process Management .....	13
1.2 The problem .....	15
1.3 Model Driven Architecture .....	17
1.4 Research questions and hypothesis .....	18
1.5 Purpose and importance of the study .....	21
1.6 Possibilities and limitations .....	23
1.7 Brief overview of the research methods and methodology used .....	24
1.8 Explanation of the structure of this work .....	25
<b>PART I: THEORETICAL BACKGROUND .....</b>	<b>27</b>
<b>CHAPTER II: BUSINESS PROCESS MANAGEMENT .....</b>	<b>28</b>
2.1 Brief history on business process management .....	28
2.2 What is business process management? .....	29
2.3 Why is important to improve business processes? .....	31
2.4 Business process management lifecycle .....	35
<b>CHAPTER III: BPMN .....</b>	<b>39</b>
3.1 What is BPMN? .....	39
3.2 BPMN basics .....	40
3.3 General uses of BPMN .....	49
3.4 More BPMN details .....	51
3.5 Simulating business processes .....	61
3.6 How BPMN fits in with UML .....	61
3.7 Example of process creation in BPMN .....	63
<b>CHAPTER IV: MODEL DRIVEN ARCHITECTURE .....</b>	<b>68</b>
4.1 What is the motivation? .....	68
4.2 Current information systems development process .....	72
4.3 Better development environments are needed .....	77
4.4 Model Driven Architecture .....	81
4.5 The MDA approach to deriving value from models .....	83

4.6 The structure and semantics of models .....	86
4.7 Basic concepts of MDA .....	88
4.8 MDA model transformation and execution.....	97
4.9 System lifecycle support in MDA.....	101
<b>CHAPTER V: METAMODELLING</b> .....	104
5.1 Problems in information system development.....	104
5.2 Language driven development .....	106
5.2.1 Abstraction.....	106
5.2.2 Integration .....	108
5.2.3 The complete solution .....	110
5.3 Features of Languages .....	111
5.4 What is a Metamodel? .....	113
5.5 Why Metamodel?.....	114
5.6 Metamodelling languages .....	115
<b>Part II: PRACTICAL STUDY AND IMPLEMENTATION</b> .....	117
<b>CHAPTER VI: RESEARCH METHOD AND METHODOLOGY</b> .....	119
6.1 Information sources .....	119
6.2 Methods, basic instruments and population of the study .....	121
6.3 Data management and elaboration.....	122
6.4 Difficulties faced during data collection.....	127
<b>CHAPTER VII: BIZAGI BPM SUITE</b> .....	129
7.1 Overview.....	129
7.2 Bizagi Modeler.....	132
7.2.1 Modeling a process in Bizagi Modeler .....	134
7.2.2 Creating a sub-process.....	140
7.2.3 Generating documentation, exporting and importing diagrams.....	142
7.2.4 Simulation .....	143
7.3 Bizagi Studio.....	147
7.3.1 Bizagi Studio user interface and the automation steps.....	148
7.3.2 Process modelling .....	151

7.3.3 Data modelling .....	151
7.3.4 Creating the user interface .....	167
7.3.5 Defining Business rules .....	178
7.3.6 Work allocation.....	190
7.3.7 Application Integration .....	206
7.4 Bizagi Engine.....	208
<b>CHAPTER VIII: MOSKITT .....</b>	<b>214</b>
8.1 MOSKitt framework .....	214
8.2 Set up and installation.....	215
8.3 Openxava .....	219
8.4 MOSKitt code generation model driven development.....	224
8.5 OXPortal environment .....	234
<b>CHAPTER IX: PRACTICAL APPLICATION IN THE ENTERPRISE .....</b>	<b>242</b>
9.1 The business process.....	242
9.2 Automating the process in Bizagi .....	242
9.2.1 Limitations.....	252
9.3 Automating the process in MOSKitt.....	253
9.3.1 Limitations.....	264
9.4 Simulation in Bizagi Process Modeler.....	266
9.5 Comparison and conclusions .....	272
<b>CHAPTER X: BUSINESS PROCESS MANAGEMENT PRACTICES IN THE TELECOMMUNICATION SECTOR IN ALBANIA .....</b>	<b>275</b>
10.1 Overview of the telecommunication sector in Albania.....	275
10.2 Business Process Management practices .....	279
10.3 Conclusions.....	282
<b>REFERENCES.....</b>	<b>284</b>
[13] Bizagi Process Modeler documentation Simulation levels, available at <a href="http://help.bizagi.com/processmodeler/en/">http://help.bizagi.com/processmodeler/en/</a> .....	285
[14] Bizagi Process Modeler documentation Simulation scenarios, available at <a href="http://help.bizagi.com/processmodeler/en/">http://help.bizagi.com/processmodeler/en/</a> .....	285

[15] BPMN introduction by Bizagi, available at  
<http://www.bizagi.com/eng/downloads/BPMNbyExample.pdf>..... 285

## LIST OF FIGURES

Figure 1: User access management process model.....	15
Figure 2: Business process management life-cycle .....	35
Figure 3: An example of a simple business process .....	45
Figure 4: An example of BPD with Pools.....	47
Figure 5: A segment of a process with Lanes .....	48
Figure 6: Part of a BPMN Business Process Diagram for an on-line auction system .....	54
Figure 7: Sub-processes and tasks .....	55
Figure 8: BPMN model of a credit application process.....	64
Figure 9: Credit application extended process.....	65
Figure 10: Verifying the applicant's information sub process.....	66
Figure 11: Verifying the applicant's information sub process updated with Automated Tasks. ....	67
Figure 12: Bizagi BPM Suite components [23] .....	130
Figure 13: Bizagi process modeler.....	133
Figure 14: Bizagi user interface explained .....	134
Figure 15: Provide the name of the Pool in Bizagi Modeler.....	136
Figure 16: Drag and drop lanes from the Palette in Bizagi Modeler .....	136
Figure 17: Drag and drop a Start event from the Palette in Bizagi Modeler.....	137
Figure 18: Diagramming the process using the Pie Menu in Bizagi Modeler .....	138
Figure 19: Connecting two diagram elements in a sequence flow in Bizagi Modeler.....	139
Figure 20: The basic diagram of the Purchase Request process .....	140
Figure 21: Transform the Quotations Task into a sub-process .....	141
Figure 22: Define the diagram for a sub-process .....	142
Figure 23: Bizagi Studio Wizard view.....	149
Figure 24: Process Wizard steps .....	150
Figure 25: Example entity in Bizagi .....	152
Figure 26: Example relationship in Bizagi .....	153
Figure 27: Example of Master and Parameter entity .....	155
Figure 28: System entities in Bizagi .....	156
Figure 29: Related Attribute relationship.....	157
Figure 30: One-to-one relationship.....	158
Figure 31: One-to-many relationship.....	159
Figure 32: Many-to-many relationship .....	160
Figure 33: Attribute types in Bizagi.....	161
Figure 34: Data Modelling, second step of the process wizard in Bizagi.....	162
Figure 35: Main process entity creation for the Purchase Requisition process .....	163
Figure 36: Adding attributed to entities in Bizagi.....	164
Figure 37: Attribute creation in Bizagi .....	165
Figure 38: Example of attributes that relate Master or Parameter entities in Bizagi .....	166

Figure 39: Attribute list window in Bizagi .....	166
Figure 40: Define forms, third step of the process wizard.....	168
Figure 41: Third step of the process wizard, define forms for the Purchase Request process.....	169
Figure 42: Data Model for the Purchase request process.....	170
Figure 43: Forms designer .....	171
Figure 44: Forms designer, adding containers in the Form .....	172
Figure 45: Naming group controls .....	173
Figure 46: Define forms, progressing with form creation.....	174
Figure 47: Define forms, editing columns in the table control .....	175
Figure 48: Define forms, completed form with all the controls.....	176
Figure 49: Define forms, properties of the control .....	177
Figure 50: Define forms, changing properties for the control Request date. ....	178
Figure 51: Use of Business rules in Bizagi .....	180
Figure 52: Data model to be used to explain how XPath works .....	181
Figure 53: Defining business rules.....	183
Figure 54: Fourth step of the process wizard, defining business rules .....	186
Figure 55: Example process to define business rules in process routing .....	186
Figure 56: Portion of the data model for the Purchase Request process.....	187
Figure 57: Expression tab .....	188
Figure 58: Create a new expression .....	189
Figure 59: Create a new expression, continued .....	190
Figure 60: Fifth step of the process wizard, define performers .....	192
Figure 61: The performer's assignment window .....	193
Figure 62: Define performers using Allocation Rules. ....	193
Figure 63: Specifying operands and operators.....	194
Figure 64: Define performers using Allocation Rules example.....	196
Figure 65: Vacation Leave Request process model .....	198
Figure 66: Vacation Leave Request process model, in define performers window.....	199
Figure 67: Define performers using allocation rules.....	199
Figure 68: Define performers for the Approve Vacation Leave Request activity .....	200
Figure 69: Define performers for the Update Payroll System activity .....	201
Figure 70: Define performers for the Update Payroll System activity, define the Entity values....	202
Figure 71: Select the Assignment Method.....	202
Figure 72: Organization in Bizagi.....	203
Figure 73: Bizagi Work Portal .....	206
Figure 74: Bizagi integration capabilities .....	207
Figure 75: Bizagi Work Portal .....	211
Figure 76: How to access the work portal in Bizagi studio .....	213
Figure 77: MOSKitt folder.....	216
Figure 78: Defining the MOSKitt workspace .....	216

Figure 79: MOSKitt environment in execution .....	217
Figure 80: Create a new project in MOSKitt .....	218
Figure 81: MOSKitt working environment.....	219
Figure 82: Create an UML2 model in MOSKitt .....	226
Figure 83: Create an UML2 model in MOSKitt, selecting the parent folder.....	227
Figure 84: UML class diagram for the Invoice Management .....	229
Figure 85: UML2JPA transformation, parameters introduction in the transformation.....	232
Figure 86: UML2JPA transformation, validating the parameters of the transformation .....	233
Figure 87: BPMN model for the user access management flow.....	244
Figure 88: The data model for the User Access Management flow.....	246
Figure 89: User interface for the Enter Permission Request activity.....	247
Figure 90: Business rules to route the User Access Management process .....	249
Figure 91: Define performers for the activity Approve Permissions LM.....	250
Figure 92: Allocation of the Corporate Security manager to the activity Approve Permissions CM. .....	251
Figure 93: User Access Management application in execution .....	252
Figure 94: UML model of the User Access Management flow. ....	254
Figure 95: MOSKitt model driven development strategy – chain of model transformations.....	256
Figure 96: Sketcher diagram for the Request Module. ....	257
Figure 97: UAM application in execution, Request module.....	257
Figure 98: Company – Address relationship, UAM model. ....	258
Figure 99: Company.java class obtained automatically through the UML2JPA transformations..	260
Figure 100: Request.java class with annotations for the user interface specified through the Sketcher model. ....	264
Figure 101: Max. arrival count for the user access management process.....	267
Figure 102: Probabilities definition for the first exclusive gateway. ....	267
Figure 103: Probabilities definition for the second exclusive gateway. ....	268
Figure 104: Resources defined for the User Access Management process.....	271
Figure 105: Simulation results for the user access management process .....	272
Figure 106: Number of Fixed Telephony subscribers per operator, end of 2014. ....	278



# **CHAPTER I: INTRODUCTION**

## **1.1 Business Process Management**

Business Process Management (BPM) is a management approach that comprises the creation, development, maintenance and optimization of business processes in different organizations. There are different definitions for a business process but they all can be summarized to as “a collection or ordered tasks that need to be carried out to achieve a certain business goal, such as the production of a particular product or the delivery of a service”. Business Process Management sees processes as strategic assets of an organization that must be understood, managed, and improved. Tasks within business processes represent activities that are performed manually by a human or automatically by some system. The main focus of BPM is the automation and optimization of business processes as well as the support of human interactions with the use of information technologies [24].

BPM is made of different phases, such as analysis, design, implementation, deployment, monitoring and evaluation. In the analysis and design phases, a business case is analyzed and a desired solution is modeled. Then the solution is implemented and executed in an existing environment (implementation and deployment phase). The running solution is controlled in the monitoring phase and the solution is adapted for continuous improvement during the evaluation phase [24].

In BPM, business processes of an organization are represented in terms of process models capturing the business logic that shall be automated. A process model comprises a set of activities connected by edges that determine the order in which the activities are performed.

Process models can be specified visually or textually. Figure 1 shows a visual process model in the Business Process Modeling and Notation (BPMN) that represents the user access management flow in modern corporations.

Business Process Model and Notation (BPMN) is a standard for business process modeling that provides a graphical notation for specifying business processes in a Business Process Diagram (BPD), based on a flowcharting technique very similar to activity diagrams from Unified Modeling Language (UML) [57]. The objective of BPMN is to support business process management, for both technical users and business users, by providing a notation that is intuitive to business users, yet able to represent complex process semantics.

The primary goal of BPMN is to provide a standard notation readily understandable by all business stakeholders. These include the business analysts who create and refine the processes, the technical developers responsible for implementing them, and the business managers who monitor and manage them. Consequently, BPMN serves as a common language, bridging the communication gap that frequently occurs between business process design and implementation.

The process model describes the necessary steps that need to be followed for an employee to obtain access into a particular system within the company. The activities in the process model are connected by edges that indicate their execution order [24].

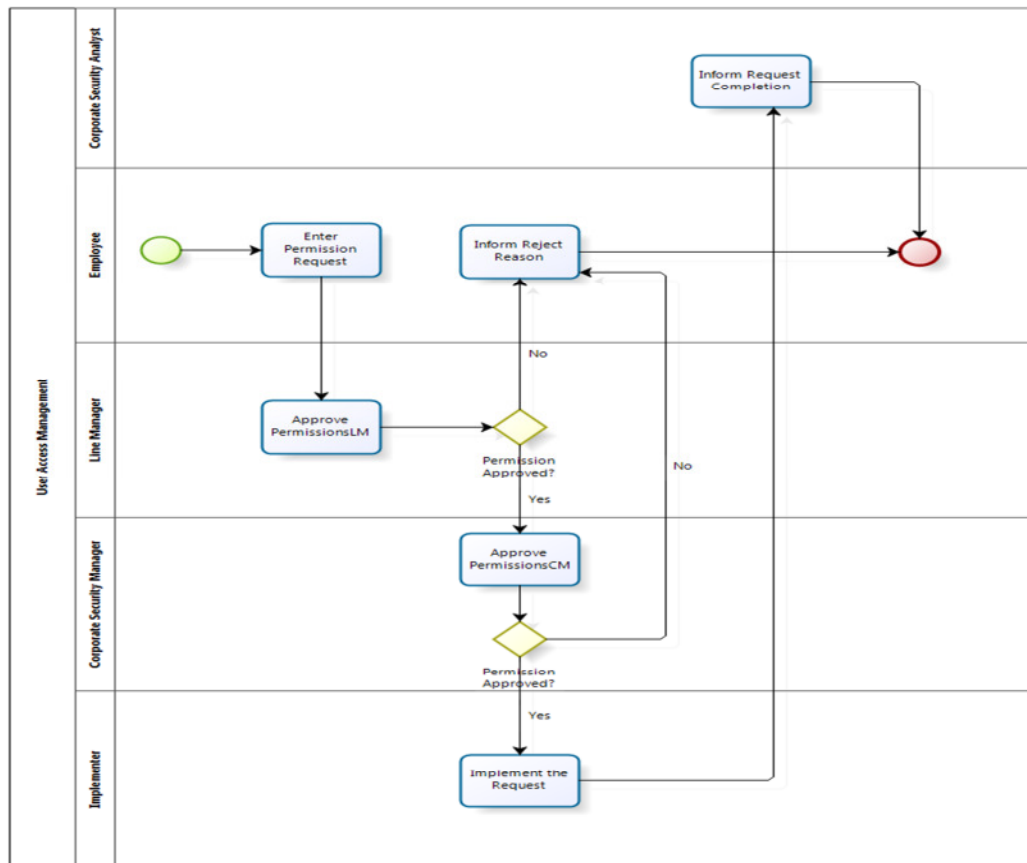


Figure 1: User access management process model

## 1.2 The problem

The automation of business processes and the development of information systems have always been and still is a complex task. The bad news is that complexity continues to increase. This due to the fact that customer and business needs are more and more sophisticated. At the same time the resulting information systems are developed and implemented using highly dynamic, constantly changing technologies [53]. In order to survive in a world always more competitive, more difficult to predict, more connected and

challenging, business environment changes dynamically. Regulatory compliance, mergers and acquisitions, joint ventures, outsourcing activities just to name a few, impose changes in the business model and therefore on the information system. At the same time the progress and introduction of new technologies impose changes in the business environment.

To create business information systems today is costly (highly paid resources over extended periods of time), too slow for modern business conditions and highly risky (difficult to control and high failure rates) [53]. The modern development process of information systems has not changed much over the past years. It is highly focused on the codification (the “how”) rather than modeling (the “what”) which affects in a negative way the productivity and the quality of the final product. Most of the efforts are spend in absorbing the technological complexity of a particular implementation rather than concentrating on understanding the business process for which the information system is being built [52, 26, 22]. Programing is still the most important task in creating a software product. It is true that programming technologies have improved with the passing of years, but the problem is that if we consider the bad results programming-oriented technologies have achieved historically is logical to ask the following: is reasonable to search for better methods of creating information systems? It would be desirable to have a new approach in developing information systems that separates business logic from the implementation technology, which takes the software development process to a higher level of abstraction using expressions that are more close to the business logic.

In this study we will focus our attention on the telecommunication sector in Albania as one of the most developed and modernized industries in the country also with the presence of

big multinational companies in operation, in order to identify what are the current Business Process Management practices implemented and provide concrete solutions to the problems the industry faces in terms of information system development and Business Process Management. The final goal is to identify a framework that would be appropriate for the telecommunication sector in Albania to solve the above problems and improve the automation and optimization of business processes and the overall information system development strategy.

### **1.3 Model Driven Architecture**

For the automation of business processes, typically the model driven architecture (MDA) to software development is applied.

In order to address the above concerns expresses in section 1.2 the Object Management Group (OMG) has defined a new framework for developing information systems, the Model Driven Architecture (MDA) [41, 42]. As per OMG definition of MDA, “*MDA separates business and application logic from underlying platform technology... No longer tied to each other, the business and technical aspects of an application can each evolve at its own pace – business logic responding to business need, and technology taking advantage of new developments – as the business requires*” [42]. At the center of the MDA approach are models; the process of software development is driven by constructing models that represent the software under development, creating software means creating the model. The specific code that represents the implementation of the model in a certain underlying technology is obtained using model transformation capabilities [52].

The MDA approach in developing information systems is a model-oriented approach since it focusses on the business logic (the “what”) rather than on the specific implementation technology (the “how”) on a particular programming environment such as Java, .NET etc [52, 23, 26]. This separation enables also portability, interoperability and reusability [45]. MDA makes models the authentic protagonist of information systems development, not the source code. This separation decreases the impact that technology evolution has on application development and also enables the possibility to profit from new implementation technologies. Once the model is created, it can be transformed automatically into code in different software platforms. Knowledge and intellectual property engaged in application development are moved from source code to models. Models are the most valuable assets since code is obtained from them through automatic transformations [53].

#### **1.4 Research questions and hypothesis**

The MDA paradigm in process automation and information system development even though not new it's not widely spread in the community of information system developers and business process analysts in Albania. Even in the most consolidated technological companies operating in the country there is not a clear focus on MDA development. The traditional way of developing information systems and automating business processes, the one that is based on specialized knowledge on a particular technology is mostly operated. Modeling and further automating a business process using a platform that is based on the model driven architecture approach would enable in a real situation the comparison between the old philosophy of process automation (the one based on programming on a particular developing environment) and the new (the one based on the model driven

architecture and process modeling). The business process automated using the approach based on models would present a more correct automation of the business process since it is focused purely on the business logic and not the implementation technology. The validity of such a claim for a real process in a modern company in Albania would be a very good start to change the mentality of the local Chief Information Officers or Information Technology Managers and try to focus their attention more on the MDA paradigm as a long term benefit for the respective companies.

Another major benefit would be the consent for the importance of Business Process Modelling in the modern companies operating in Albania. The environment in which the modern companies operate is highly dynamic. They use a set of technologies to try to reduce their costs but few of them use formal methods and models to create, manage and communicate business process models. This is also due to the fact that developments in information technology has been focused on data modeling and mostly due to the special focus for the complexity represented from a particular technology than the correct understanding of the business problem to be solved.

Most of the investments in information technology that can be performed from a company not necessarily can be a competitive advantage for that company. Let suppose that all competitors in a particular market and industry (e.g. telecommunication industry in Albania) purchase and implement a particular information system in order to streamline their processes and improve costs and performance (e.g. a Customer Relationship Management –CRM – system). Let consider that there is a leader in the CRM systems market, CRM X. All telecommunication companies can purchase and implement the CRM X system. There are very few chances that a particular company to have a competitive

advantage compared to the others by operating in this way. The capability to purchase the best CRM – X system cannot be considered a competitive advantage since competitors can purchase that system as well. The company that would best benefit would be the one that has a clear picture and total understanding of the business process to be automated and then automates the respective process using a particular information technology solution. In order to achieve success and outperform the competitors a modern company should look further the particular information technology solution and consider in a broader view all the business processes.

There has been a big debate in the industry. The question is: “Do information technology investments pay off?” The answer to such a debating question finally is clear: “Yes, information technology investments pay off if there are clear and sound business processes in place within the company”. If the company takes the journey to automate a particular business process without a clear and total understanding itself of the business process, the journey will be a long, costly and not enjoyable. Chances are that the new information technology system implemented within the company will not provide more clarity but will further complicate the situation. There are many cases where large projects have failed and in most cases the failure of such projects can be attributed to the lack of clear business processes in place.

So what does it mean to have clear processes in place? The best way to understand a process is to model it in a particular modeling formalism. Would be desirable to have a modeling formalist not too technical and complicated but one that is easy to understand, immediate to read and intuitive. That way the model would be easily shared with all stakeholders within the company involved with the process from technology, finance,



human resources and business colleagues. Having such a formalism that is easily understandable from all stakeholders within the company is one of the main research areas we will consider. We consider Business Process Modelling as a strong differentiator for modern companies and as the ultimate tool which provides competitive advantage to them. The research questions upon which we will focus our attention in this study are as follows. What is the state of Business Process Management in the telecommunication sector in Albania and if model oriented approaches are used in process automation? Is business process simulation considered to optimize business process in the industry? What would be an appropriate framework that is suitable for the industry to provide a complete support for Business Process Management and process automation? Our hypothesis are that Business Process Management practices in the telecommunication sector in Albania should be present but not well structured and organized according to international standards, not clear formalisms should be in place to model business processes and we believe that process automation is not model oriented.

A successful application in Business Process Modelling through the usage of the Model Driven Architecture approach in automating the particular business process in an existing and important company in Albania would potentially create the opportunity to open the Business Process Modelling consultancy and process automation market in Albania.

### **1.5 Purpose and importance of the study**

Considering the complexity in which modern companies operate and the frequent technological changes “where the only constant is change” the purpose of this study is to provide a complete overview of Business Process Management and Model Driven

Architecture approach in process automation in and information system development in the telecommunication sector in Albania through the modeling and automation of real processes. The real case successful application in a particular framework of business process automation based on models would clearly identify a possible suite which is suitable for the Business Process Management community and also the benefits of this model oriented approach in information system development and process automation will be evidenced.

In the first part is provided a complete overview background on Business Process Modelling and Model Driven Architecture. In the business process modelling arena different modeling formalisms will be presented and the main focus will be Business Process Modelling and Notation (BPMN) which is the standard in Business Process Modelling. Further the focus will be on the Model Driven Architecture approach as the natural application of the automation of business processes.

In the second part we will focus our attention in two different solution implementations of the MDA approach in process automation which will be considered in detail and will be used in order to automate business processes.

- MOSKitt Framework
- Bizagi business process management suite

The same business process will be automated using both solutions highlighting the pros and cons of each one and suggesting which solution is more suitable in each case. In the last part it will be presented a complete overview of the application of Business Process Management practices within the telecommunication sector in Albania.

The successful completion of the study provides a clear example on the application of the model oriented approach in process automation and would contribute in raising awareness in the local business community on the benefits of Business Process Modelling and Process Automation using this paradigm. The comparison of two different model oriented implementations is also a particular contribute in order to facilitate the decision for which solution is best in each case. Furthermore once again the validity of the model driven architecture approach as the future in developing information systems would be clearly evidenced. The introduction of the importance of Business Process Modelling to the Albanian market and presentation of state of the art Business Process Management suites would contribute in providing clear competitive advantages to modern companies operating in Albania to outperform their competitors that in a world always more connected and competitive cannot be only local but also global ones.

## **1.6 Possibilities and limitations**

The model driven architecture approach in information system development and process automation represents a great opportunity towards a further important step in the process of information system development, jumping from the current development process based on programming at a more abstract level which is based on the model that represents the logic of the business process and is independent from the underlying implementation technology. The MDA paradigm in software development is considered as the biggest change since the shift from Assembler to the high level programming languages that are used today like Java, C, C++ etc [24].

The fact that the MDA paradigm and Business Process Modelling practices are not popular in the local community can represent challenges regarding the successful modelling and automation of a real business process. The lack of local information and support on the matter needs to be taken into consideration, since possible issues faced during the development might not be easy and immediate to be solved. In some cases the transformation from the model to the respective automated information system is not complete meaning that there is still the need to intervene manually and fill some parts of the code. There is still the needed for manual programming in some cases and this is related to the particular state of development of an MDA implementation.

The process upon which we will focus our attention will be an existing process in a telecommunication company. The User Access Management processes that employee's within the company follow in order to gain access into a particular system. Further and more complex processes can be analyzed but are not considered in this work. They can be considered for further developments.

### **1.7 Brief overview of the research methods and methodology used**

It will be a first practical application part which consists on the automation of the business process model into two different process automation suites. In this case the attention will be focused on the study and comprehending of the methods in order the above suites in operation and fully functional. Once the both suites are functional locally the research will be focused on automating a particular business process in a modern company in Albania using both suites. In the first stage there will be performed several interviews with the business stakeholders of the User Access Management process to be automated in order to

fully comprehend the process and its operation. After the interview phase and when the process is clear we will process with process modelling and process automation in the respective suites. A comparison study between the two different suites will be performed in order to suggest which the best alternative is in each specific case.

In the second part will be used qualitative methods in order to evidence the state of the art in the Business Process Management field in the telecommunication industry in Albania. Questionnaires and interviews will be used with respective stakeholders for Business Process Management in all the major telecommunication companies operating in Albania and which are worth considering for the study.

### **1.8 Explanation of the structure of this work**

This work is structured in two fundamental parts. Part one is focused on the theoretical background while the second part is focused on the practical applications of the study.

In the first part we firstly focus our attention on explaining what Business Process Management is and why it is important by providing a complete overview of the debate in this field and further explaining the steps of the Business Process Management lifecycle. Then we justify the importance of modelling formalisms to be used for business process modelling and in particular the details of the Business Process Modelling and Notation (BPMN) standard are presented. Also a brief overview of Unified Modelling Language (UML) is presented. This first theoretical part is completed by a deep analysis and presentation of the Model Driven Architecture approach in process automation presenting the reasons and advantages and we conclude the section by discussing the importance of metamodeling.

In the second part we present two different frameworks that we have used for process automation. The process that will be considered to be automated is the User Access Management process and this process is modelled and automated in both frameworks Bizagi BPMS suite and MOSKitt. The details on how to automate the process in each framework are presented and finally a comparative study on the pros and cons of each solution is performed in order to evidence which one is more applicable in each case. This is a fundamental part of the practical applications of the study since enables us to suggest that Bizagi BPMS suite is a framework suitable for process automation and optimization for the telecommunications industry in Albania. We complete the second part by providing a complete overview of the Business Process Management practices in the telecommunication industry in Albania. The results demonstrate that Bizagi BPMS suite would be a very good match for business process management in the respective companies.

## **PART I: THEORETICAL BACKGROUND**

The first part of this work is dedicated to a complete theoretical background and literature review on Business Process Management, Business Process Modelling and Notation (BPMN) and Model Driven Architecture. A detailed literature review is performed for each area in order to provide a comprehensive understanding of each topic and to present the different ways of thinking present in each of them. Firstly we focus our attention on explaining what Business Process Management is and why it is important by providing a complete overview of the debate in this field and further explaining the steps of the Business Process Management lifecycle. Then we justify the importance of modelling formalisms to be used for business process modelling and in particular the details of the Business Process Modelling and Notation (BPMN) standard are presented. Also a brief overview of Unified Modelling Language (UML) is presented. This first theoretical part is completed by a deep analysis and presentation of the Model Driven Architecture approach in process automation presenting the reasons and advantages and we conclude the section by discussing the importance of metamodeling.

## **CHAPTER II: BUSINESS PROCESS MANAGEMENT**

### **2.1 Brief history on business process management**

As far back as 1911, Frederick Taylor focused on manufacturing tasks and time/motion studies, which were measured statistically. In order to maximize profits, the primary business drivers were efficiency and minimized cost. An organization would focus on training its workers to follow specific steps that required narrowly focused skill and endurance. Standards and controls were mechanistic. Process drivers were maximized around distinct, insular, repeatable tasks. However, given the business environment at that time, the business areas were intentionally siloed [29].

In the 1960s, technology increasingly became a business driver and amplified the speed of change. This launched the first wave of process orientation. International (Japanese) companies became much more competitive, due, in part, to their focus on quality improvement programs and reduced defects. US companies started to mirror the quality approach. The combination of process scrutiny and technological superiority led to technology as process driver. American business changed its operational paradigm, and the process era began [29].

American business scrutiny of international competition changed focus to measurable processes and to speed that could be combined into “Just in time” manufacturing. The growing use of computers in the 1970s and 80s combined with procedure specialization that accommodated technological precision in fields such as nuclear power, and led to quantitative statistical software and related data gathering techniques that measured, gathered, and interpreted results [29].



The second wave of process orientation covered the late 1980s to the early 1990s. Revenue growth returned as American companies leveraged international process practices. Focus shifted to TQM, and then to ISO compliance standards. Over a decade of statistical analysis increased the need to manage data in a meaningful way. The organization shifted from a focus on corporate mission and group brainstorming to cross-functional teams and to handoffs within the organization as the “how” to do tasks replaced the “why [29].”

The third wave began in the mid-1990s and continues in the present as the “coming of age” of process-centric business. Technology is shifting from being a process driver to a process enabler. The identity of the customer changed from markets to individuals with customized solutions. Just-in-time manufacturing of the first wave led to just-in-time supply chains of the third wave, with the accompanying need to understand processes across disparate enterprises. The company as a system became more important than an examination of its individual parts. With the advent of thin-client applications and commonly used protocols, applications could be utilized regardless of the operating system or work station. This allowed “business management” to start to separate from “systems management” and enabled “process management” to exist separate from the systems themselves [29].

## **2.2 What is business process management?**

There are as many answers to this question as there are vendors, analysts, researchers, academics, commentators and customers. BPM does not equate to a technology tool or initiative for business processes. Based on the experience, there is significant business process improvement that can be achieved without technology [28].

The following question would be obvious. Can BPM involve technology, and is technology a good thing? Absolutely, in the right circumstances and when it can be justified. Are process modeling and management tools useful for achieving process improvements in non-technology circumstances? If the tools referred to are process-modeling tools, then yes, they can be extremely useful in this process [28].

The below is a common myth believed today. There is a constant danger of organizations believing that once they have purchased a process-modeling tool, it will solve all their problems and the process improvements will just follow. Nothing could be further from the truth. A process-modeling tool is just some software, and without a methodology, skilled people and resources to use it and a clear commitment from top management, it is of no use [28].

Nowadays many of the industry players and vendors provide definitions that specify technology (automation tools) as an essential component of BPM – in fact, they say that BPM is technology. However, if we take a simple and commonsense view of BPM, it is obviously about the management of business processes. If we consider this simple statement in mind and keep the organization the primary focus, we would suggest that BPM is: *The achievement of an organization's objectives through the improvement, management and control of essential business processes* [28].

Process management is an integrated part of 'normal' management. It is important for leadership and management to recognize that there is no finish line for the improvement of business processes; it is a continuous program that must be continually maintained.

In a summary we could say that BPM is:

- more than just software

- more than just improving or reengineering your processes – it also deals with the managerial issues
- not just hype – it is an integral part of management
- more than just modeling – it is also about the implementation and execution of these processes, which requires analysis.

Last but not least, as a management discipline BPM requires an end-to-end organizational view and a great deal of common sense, both of which can often be in short supply [28].

### **2.3 Why is important to improve business processes?**

*The first rule of any technology is that automation applied to an efficient operation will magnify the efficiency.*

*The second is that automation applied to an inefficient operation will magnify the inefficiency.*

(Bill Gates)

It is part of the human nature to be attracted to easy solutions for complex problems. In the business world we learn to solve problems quickly and move on fast. In the case when we can't get our work done quickly enough, we've discovered we can automate! In the office we've perfected this ability over the past 100 years to the point that we now automate almost everything in order to get more done faster – it began first with letter writing, then book-keeping, reporting, inventory, sales and order processing and, more recently, business workflow and document management. Thus, when confronted with productivity, efficiency

and business control issues today, our first temptation is to buy an automated solution to our problem [28].

Businesses today, especially very large organizations with complex service products, are realizing that there's only so much their IT systems can achieve in improving their business operations. Even where core systems are effective and efficient (which is not always the case), it is becoming increasingly difficult further to improve the overall operating efficiency and customer service effectiveness to the extent necessary to meet customer and shareholder expectations at a rate faster than our competitors. Even Bill Gates, the ultimate advocate of technology, notes that automation is only effective when applied to efficient operations [28].

So, having solved the immediate challenges of automating operational business systems and having achieved most of the 'easy' systems benefits, organizations are now turning to some of the more difficult and systemic operational efficiency areas to achieve their required 'step change' business improvement benefits: the operational back-office processes themselves. What is our intuitive solution to these age-old areas of inefficiency? Automate them! After all, it worked for systems throughput and productivity, and there are now plenty of vendors keen to provide automated workflow and document imaging solutions – often 'out of the box' – for your industry or environment [28].

There are two main reasons why the above isn't working. The first is where executives and management see their processes as a 'black box'. They don't know the details, but somehow the processes produce outcomes. The executives have a feeling that these processes may not be as efficient or as effective as they could be, but at least they work, and managers are afraid to change anything because change might disrupt these fragile

‘black box’ processes – and fixing a problem is tough when you do not understand it.

Automating the ‘black box’ is therefore easier, because it becomes a project and businesses ‘do’ projects [28].

The second reason is where the processes and associated people are treated like sacred objects: executives cannot or do not want to discuss the efficiency and effectiveness or ask the tough questions. They keep ‘looking at the edges’ of the problem and not at the heart – solving symptoms rather than the cause. For these organizations, bringing in a new technology sounds so much easier because there is no talk about people or processes – just technology [28].

If business process inefficiencies could be easily solved by automating them, why are consultants often called in after an organization has purchased an expensive automated workflow solution that has failed to ‘solve’ the problem? Why do automated solutions fail to deliver their expected business benefits? In fact, often organizations experience an increase in paper work or increased rework and diminished quality following automation of key business processes and workflows [28].

Automating something doesn’t fix its underlying problems; it just helps them to occur more quickly, in vastly increased numbers and at greater frequency. The notion that ‘we’re going to replace what’s broken with something much better’ is almost never realized in a mature organization, owing to the difficulty of making instant process and cultural change on a broad scale while still running the business. At best a compromise solution is achieved, often following sizable project over-runs in time and cost budgets. At worst, the project fails completely and the status quo is maintained. In both cases, the expected benefits are not realized and employee and customer satisfaction levels may decline dramatically [28].

The obvious question would seem to be, '*Why don't organizations fix their processes before they look to automation solutions?*' .In most large organizations, the basic back-office processes have remained predominantly unchanged for many years – even decades. In the financial services industry, for example, basic banking, insurance and investment processing procedures have been passed down for generations.

Historically, organizations haven't been able to fix their operational back office (business processes) easily because they are perceived as being either easy (and all we need to do is automate them to make them faster and take people out of the equation as much as possible) or hard (and too difficult for management to fix because they do not have the expertise, and the temptation is to purchase a solution which will 'solve' the problem – so back to the easy option) [28].

Management, at the operational level, is predominantly about the improvement and control of the processes essential to your business to achieve the objectives of the organization.

Setting the direction and goals for business process improvement is a critical step, and one that needs to be addressed by higher management. While the introduction of technology can be a useful contributor for many organizations, business process improvement does not always need technology to be successful. It is far more important to get your processes right before you consider the implementation of technology. It is executive management's responsibility to ensure there is a clear link between the process improvement projects undertaken by the business, and the organization's strategy and objectives. If the project cannot make a strong case on how it contributes and adds value to an organization's objectives, the project should not be undertaken [28].

## 2.4 Business process management lifecycle

Business process management activities can be arbitrarily grouped into categories such as design, modeling, execution, monitoring, and optimization as presented in Figure 2:

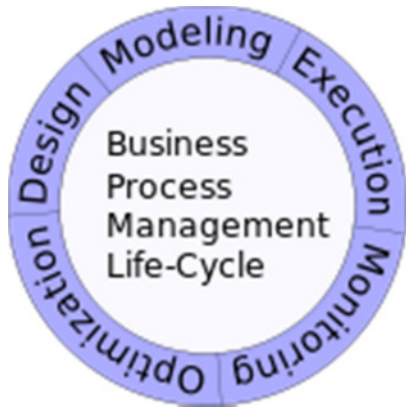


Figure 2: Business process management life-cycle

### Design

Process design encompasses both the identification of existing processes and the design of "to-be" processes. Areas of focus include representation of the process flow, the factors within it, alerts and notifications, escalations, standard operating procedures, service level agreements, and task hand-over mechanisms. Whether or not existing processes are considered, the aim of this step is to ensure that a correct and efficient theoretical design is prepared [17].

The proposed improvement could be in human-to-human, human-to-system or system-to-system workflows, and might target regulatory, market, or competitive challenges faced by

the businesses. The existing process and the design of new process for various applications will have to synchronize and not cause major outage or process interruption [17].

### **Modeling**

Modeling takes the theoretical design and introduces combinations of variables (e.g., changes in rent or materials costs, which determine how the process might operate under different circumstances). It may also involve running "what-if analysis"(Conditions-when, if, else) on the processes: "*What if I have 75% of resources to do the same task?*" "*What if I want to do the same job for 80% of the current cost?*" [17].

### **Execution**

One of the ways to automate processes is to develop or purchase an application that executes the required steps of the process; however, in practice, these applications rarely execute all the steps of the process accurately or completely. Another approach is to use a combination of software and human intervention; however this approach is more complex, making the documentation process difficult.

As a response to these problems, software has been developed that enables the full business process (as developed in the process design activity) to be defined in a computer language which can be directly executed by the computer. The system will either use services in connected applications to perform business operations (e.g. calculating a repayment plan for a loan) or, when a step is too complex to automate, will ask for human input. Compared to either of the previous approaches, directly executing a process definition can be more straightforward and therefore easier to improve. However, automating a process definition requires flexible and comprehensive infrastructure, which typically rules out implementing these systems in a legacy IT environment [17].



Business rules have been used by systems to provide definitions for governing behavior, and a business rule engine can be used to drive process execution and resolution.

### **Monitoring**

Monitoring encompasses the tracking of individual processes, so that information on their state can be easily seen, and statistics on the performance of one or more processes can be provided. An example of this tracking is being able to determine the state of a customer order (e.g. order arrived, awaiting delivery, invoice paid) so that problems in its operation can be identified and corrected. In addition, this information can be used to work with customers and suppliers to improve their connected processes. Examples are the generation of measures on how quickly a customer order is processed or how many orders were processed in the last month. These measures tend to fit into three categories: cycle time, defect rate and productivity.

The degree of monitoring depends on what information the business wants to evaluate and analyze and how business wants it to be monitored, in real-time, near real-time or ad hoc. Here, business activity monitoring (BAM) extends and expands the monitoring tools generally provided by BPMS [17].

Process mining is a collection of methods and tools related to process monitoring. The aim of process mining is to analyze event logs extracted through process monitoring and to compare them with an a priori process model. Process mining allows process analysts to detect discrepancies between the actual process execution and the a priori model as well as to analyze bottlenecks [17].

### **Optimization**

Process optimization includes retrieving process performance information from modeling or monitoring phase; identifying the potential or actual bottlenecks and the potential opportunities for cost savings or other improvements; and then, applying those enhancements in the design of the process. Overall, this creates greater business value [17].

## **CHAPTER III: BPMN**

### **3.1 What is BPMN?**

BPMN stands for Business Process Modeling Notation. It is the new standard for modeling business processes [50]. Business Process Management Initiative (BPMI) developed BPMN, which has been maintained by the Object Management Group since the two organizations merged in 2005. As of March 2011, the current version of BPMN is 2.0 [16]. With the version change from BPMN to BPMN 2.0 the name has been adapted to Business Process Model and Notation as beginning with version 2.0 the language does not only contain notational information, but execution semantics.

The primary goal of the BPMN effort is to provide a notation that is readily understandable by all business users, from the business analysts who create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and, finally, to the business people who will manage and monitor those processes. A second, equally important goal is to ensure that XML languages designed for the execution of business processes, such as BPEL4WS (Business Process Execution Language for Web Services) and BPML (Business Process Modeling Language), can be visually expressed with a common notation. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation [16, 58, 50].

BPMN defines a Business Process Diagram (BPD), which is based on a flowcharting technique tailored for creating graphical models of business process operations. A Business

Process Model, then, is a network of graphical objects, which are activities (i.e., work) and the flow controls that define their order of performance [58].

*“To improve is to change; to be perfect is to change often.” -- Winston Churchill*

BPMN is a core enabler for a Business Process Management (BPM). BPM is concerned with managing change to improve business processes. BPM is unifying the previously distinct disciplines of Process Modeling, Simulation, Workflow, Enterprise Application Integration (EAI), and Business-to-Business (B2B) integration into a single standard. The fact that Business Process Management is a new initiative might lead you to believe that business processes have not been managed previously. This is of course not true – many organizations have modeled and managed their business processes for years, using an eclectic mixture of tools and techniques. These techniques have only been partially successful, or failed outright, because there has been a lack of standards and a complete lifecycle to control and guide the design and execution of business processes. Managing the process of change cannot be an ad-hoc process – it requires management to exercise control over the discovery, architecture, design, and deployment of processes. For management to understand the architecture, design, and deployment of processes, is needed business modeling and business execution language standards [50].

### **3.2 BPMN basics**

BPMN is constrained to support only the concepts of modeling applicable to business processes. Other types of modeling done by organizations for non-process purposes are out of scope for BPMN. Examples of modeling excluded from BPMN are:

- Organizational structures

- Functional breakdowns
- Data models

A BPD is made up of a set of graphical elements. These elements enable the easy development of simple diagrams that will look familiar to most business analysts (e.g., a flowchart diagram). The elements are chosen to be distinguishable from each other and to utilize shapes that are familiar to most modelers. For example, activities are rectangles, and decisions are diamonds. It should be emphasized that one of the drivers for the development of BPMN is to create a simple mechanism for creating business process models, while at the same time being able to handle the complexity inherent to business processes. The approach taken to handle these two conflicting requirements was to organize the graphical aspects of the notation into specific categories. This provides a small set of notation categories so that the reader of a BPD can easily recognize the basic types of elements and understand the diagram. Within the basic categories of elements, additional variation and information can be added to support the requirements for complexity without dramatically changing the basic look-and-feel of the diagram [16, 58]. The four basic categories of elements are:

- Flow Objects
- Connecting Objects
- Swimlanes
- Artifacts

### **Flow Objects**

A BPD has small set of (three) core elements, which are the Flow Objects, so that modelers do not have to learn and recognize a large number of different shapes. The three Flow Objects are [58]:

An *Event* is represented by a circle and is something that “happens” during the course of a business process. These Events affect the flow of the process and usually have a cause (trigger) or an impact (result). Events are circles with open centers to allow internal markers to differentiate different triggers or results.

#### **Event**

There are three types of Events, based on when they affect the flow: *Start*, *Intermediate*, and *End* (see the figures to the right, respectively). Events are also classified as *Catching* (for example, if catching an incoming message starts a process) or *Throwing* (such as throwing a completion message when a process ends).



### **Activity**

An *Activity* is represented by a rounded-corner rectangle (see the figure to the right) and is a generic term for work that company performs. An Activity can be atomic or non-atomic (compound). The main types of Activities are: *Task* and *Sub-Process*. The Sub-Process is distinguished by a small plus sign in the bottom center of the shape.



### **Gateway**

A *Gateway* is represented by the familiar diamond shape (see the figure to the right) and is used to control the divergence and convergence of Sequence Flow. Thus, it will determine traditional decisions, as well as the forking, merging, and joining of paths. Internal Markers will indicate the type of behavior control.



### **Connecting Objects**

The Flow Objects are connected together in a diagram to create the basic skeletal structure of a business process. There are three Connecting Objects that provide this function. These connectors are [16, 58]:

**Sequence Flow** *A Sequence Flow* is represented by a solid line with a solid arrowhead (see the figure to the right) and is used to show the order (the sequence) that activities will be performed in a Process. Note that the term “control flow” is generally not used in BPMN.



**Message Flow** *A Message Flow* is represented by a dashed line with an open arrowhead (see the figure to the right) and is used to show the flow of messages between two separate Process Participants (business entities or business roles) that send and receive them. In BPMN, two separate Pools in the Diagram will represent the two Participants.





An *Association* is represented by a dotted line with a line arrowhead (see the figure to the right) and is used to associate data, text, and other Artifacts with flow objects. Associations are used to show the inputs and outputs of activities.



For modelers who require or desire a low level of precision to create process models for documentation and communication purposes, the core elements plus the connectors will provide the ability to easily create understandable diagrams as shown in Figure 3 [58]:

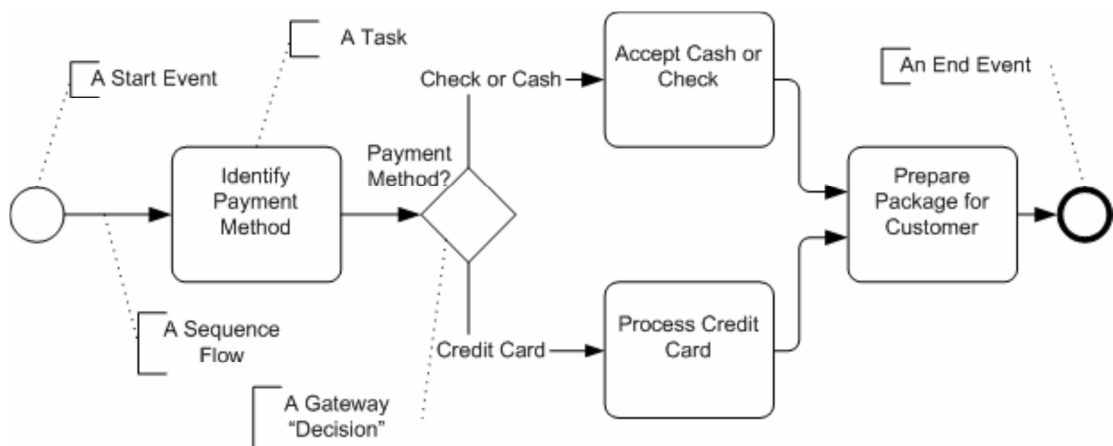


Figure 3: An example of a simple business process

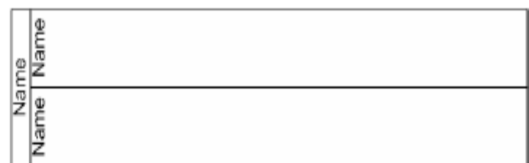
## Swimlanes

Many process modeling methodologies utilize the concept of swimlanes as a mechanism to organize activities into separate visual categories in order to illustrate different functional capabilities or responsibilities. BPMN supports swimlanes with two main constructs. The two types of BPD swimlane objects are:

**Pool** A *Pool* represents a Participant in a Process. It is also acts as a graphical container for partitioning a set of activities from other Pools (see the figure to the right), usually in the context of B2B situations.



**Lane** A *Lane* is a sub-partition within a Pool and will extend the entire length of the Pool, either vertically or horizontally (see the figure to the right). Lanes are used to organize and categorize activities.



Pools are used when the diagram involves two separate business entities or participants (see Figure 4) and are physically separated in the diagram. The activities within separate Pools are considered self-contained Processes. Thus, the Sequence Flow may not cross the boundary of a Pool. Message Flow is defined as being the mechanism to show the communication between two participants, and, thus, must connect between two Pools (or the objects within the Pools) [16, 58].

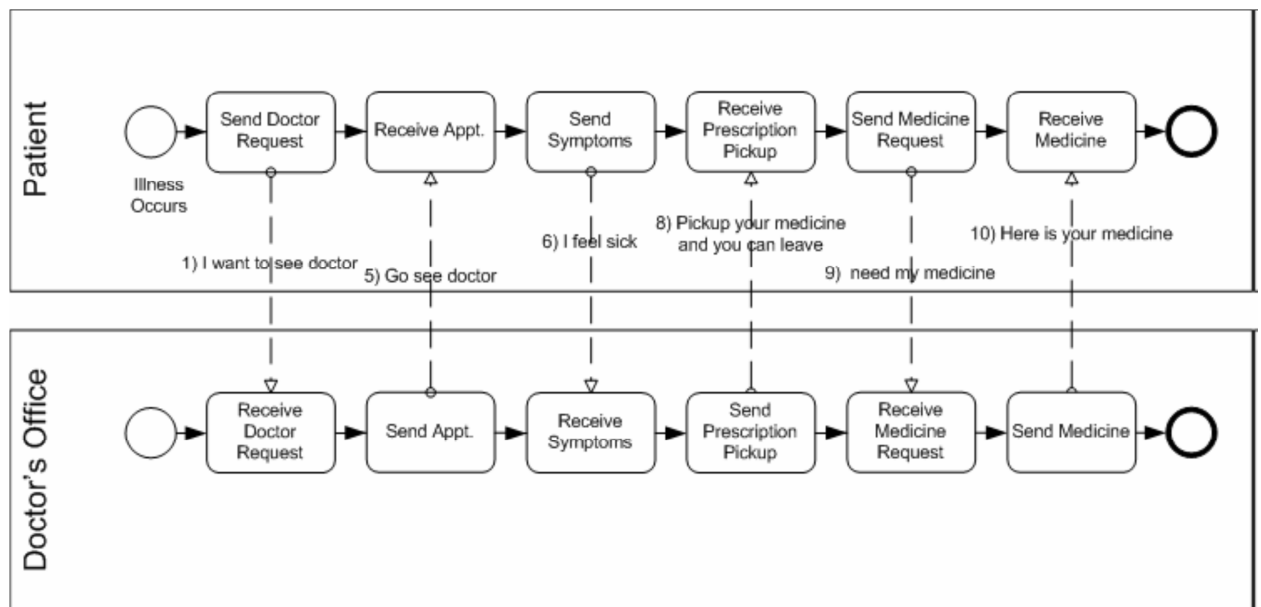


Figure 4: An example of BPD with Pools

Lanes are more closely related to the traditional swimlane process modeling methodologies. Lanes are often used to separate the activities associated with a specific company function or role (see Figure 5). Sequence Flow may cross the boundaries of Lanes within a Pool, but Message Flow may not be used between Flow Objects in Lanes of the same Pool [58].

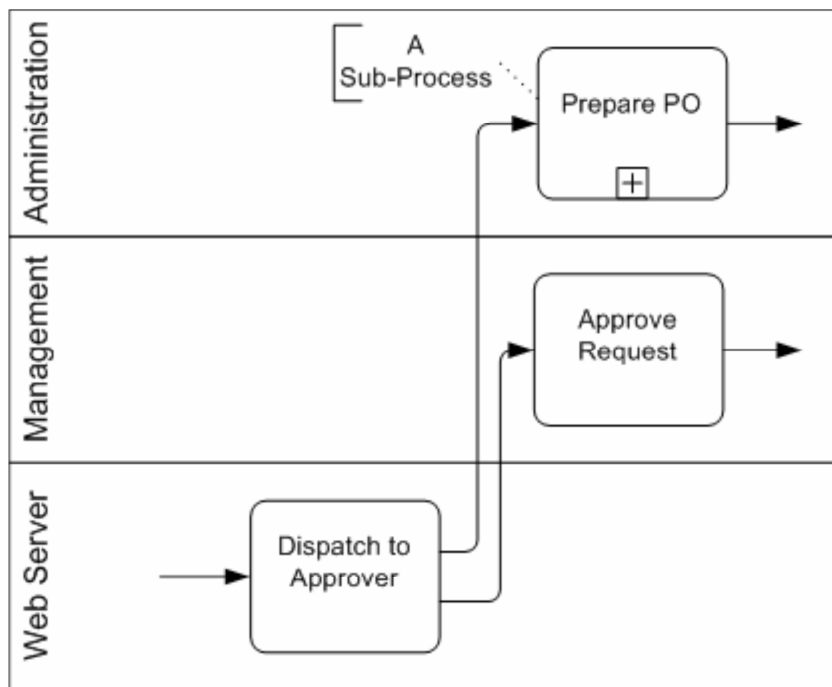


Figure 5: A segment of a process with Lanes

## Artifacts

BPMN was designed to allow modelers and modeling tools some flexibility in extending the basic notation and in providing the ability to add context appropriate to a specific modeling situation, such as for a vertical market (e.g., insurance or banking). Any number of Artifacts can be added to a diagram, as appropriate for the context of the business processes being modeled. The three main types of BPD Artifacts are [58]:

### Data Object

*Data Objects* are a mechanism to show how data is required or produced by activities. They are connected to



activities through Associations.

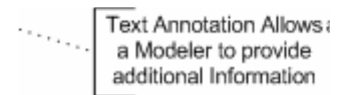
### **Group**

A *Group* is represented by a rounded corner rectangle drawn with a dashed line (see the figure to the right). The grouping can be used for documentation or analysis purposes, but does not affect the Sequence Flow.



### **Annotation**

*Annotations* are a mechanism for a modeler to provide additional text information for the reader of a BPMN Diagram (see the figure to the right).



## **3.3 General uses of BPMN**

Business process modeling is used to communicate a wide variety of information to different audiences. BPMN is designed to cover many types of modeling and allows the creation of process segments as well as end-to-end business processes, at different levels of fidelity. Within the variety of process modeling objectives, there are two basic types of models that can be created with a BPD [58]:

- Collaborative (Public) B2B Processes
- Internal (Private) Business Processes

## **Collaborative B2B Processes**

A collaborative B2B process depicts the interactions between two or more business entities. The diagrams for these types of processes are generally from a global point of view. That is, they do not take the view of any particular participant, but show the interactions between the participants. The interactions are depicted as a sequence of activities and the message exchange patterns between the participants. The activities for the collaboration participants can be considered the “touch-points” between the participants; thus, the process defines the interactions that are visible to the public for each participant. When looking at the process shown in only one Pool (i.e., for one participant), public process is also called an abstract process. The actual (internal) processes are likely to have more activities and detail than what is shown in the collaborative B2B processes. The process in Figure 4 is an example of a collaborative B2B process [58].

## **Internal business processes**

An internal business process will generally focus on the point of view of a single business organization. Although internal processes often show interactions with external participants, they define the activities that are not generally visible to the public and are, therefore, private activities. If swimlanes are used, then an internal business process will be contained within a single Pool. The Sequence Flow of the Process is therefore contained within the Pool and cannot cross the boundaries of the Pool. Message Flow can cross the Pool boundary to show the interactions that exist between separate internal business

processes. Thus, a single Business Process Diagram may show multiple private business processes [58].

### 3.4 More BPMN details

During business process modeling, you model the events that happen in the business, and show how they affect process flows. An event either kicks off a process flow, or happens during a process flow, or ends a process flow. BPMN provides a distinct notation for each of these types of events, as shown in the table below [50].

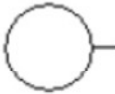


Start Event		Intermediate Event		End Event	
Start Event	Starts a process flow.	Event	Happens during the course of a process flow.	End Event	Ends a process flow.
					

Table 1: Basic event types in BPMN and their notation

When you model more complex process flows, such as B2B web services, you need to model more complex business events, such as messages, timers, business rules, and error conditions. BPMN enables you to specify the trigger type of the event, and denote it with a representative icon, as specified in Table 2. Specifying a trigger type to an event puts certain constraints on the process flow that you are modeling, which are explained in the table. For example, a timer cannot end a process flow. You can only draw message flows from and to message events. These types of modeling rules, which are actually kinds of

business rules, should be enforced automatically by the modeling tool providing support for BPMN [50].
















Start Events	Intermediate Events	End Events	Description
Start Message 	Message 	End Message 	A start message arrives from a participant and triggers the start of the process, or continues the process in the case of an intermediate event. An end message denotes a message generated at the end of a process.
Start Timer 	Timer 	A Timer cannot be an End Event.	A specific time or cycle (for example every Monday at 9am) can be set to trigger the start of the process, or continue the process in the case of an intermediate event.
Start Rule 	Rule 	A Rule cannot be an End Event.	Triggers when the conditions for a rule become true, such as "Stock price changes by more than 10% since opening."
Start Link 	Link 	End Link 	A link is a mechanism for connecting the end event of one process flow to the start event of another process flow.
Start Multiple 	Multiple 	End Multiple 	For a start multiple event, there are multiple ways of triggering the process, or continuing the process in the case of the intermediate event. Only one of them is required. The attributes of the event define which of the other types of triggers apply. For end multiple, there are multiple consequences of ending the process, all of which will occur (for example, multiple messages sent).
An Exception cannot be a Start event.	Exception 	End Exception 	An end exception event informs the process engine that a named error should be generated. This error will be caught by an intermediate exception event.



Table 2: Event trigger types

### **Business Processes, Sub-Processes, and Tasks**

At the core of business process modeling are the processes themselves. There are three types of processes – *the process*, *the sub-process*, and *the task*. Each is graphically depicted by the same rounded rectangular symbol; the use of different nouns simply reflects the hierarchical relationships between them [16, 50].

A process is a network of ‘doing things’. You draw it as a rounded rectangle on your top-level BPMN Business Process diagram. You can specify the inner details of a process by creating or attaching another Business Process diagram to it. The sub-diagram is considered a 'child' diagram. A process that has a child diagram gets a '+' marker in its body.

Graphically showing the details of a process with another Business Process diagram is considered 'decomposing' the process. You can continue to decompose a process without any restriction -- creating a child diagram for a process, and child diagrams for the processes on the first child diagram, and so forth. Processes that you draw on 'child' diagrams are considered sub-processes. The lowest-level process, which you do not decompose further, is considered a task [16, 50].

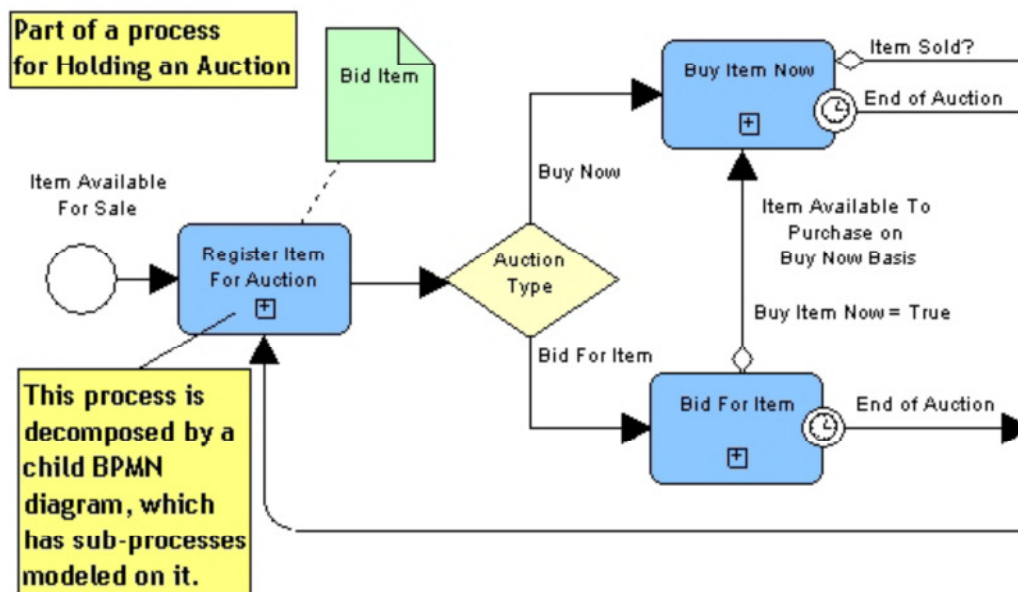


Figure 6: Part of a BPMN Business Process Diagram for an on-line auction system

Figure 6 shows a BPMN business process diagram on which the process Register Item for Auction has been modeled. The '+' mark in the process's body tells you that there is at least one 'child' business process diagram hyperlinked to this process, and on that diagram is a graphical depiction of the details of this process. Figure 7 shows part of the 'child' BPMN Business Process diagram to the Register Item for Auction process. Since they are on a 'child' diagram, the processes are considered sub-processes. Processes on this diagram that are not further decomposed (no '+' mark in their center) are considered tasks. As you can see, it is easy to pick out a task on a diagram – simply those rounded rectangles without a '+' mark at their center.

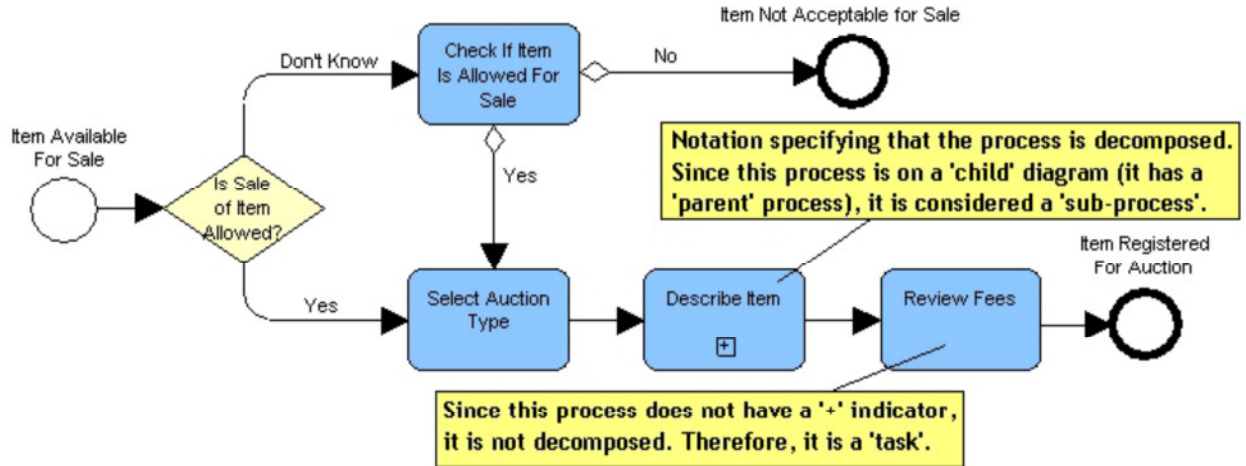


Figure 7: Sub-processes and tasks


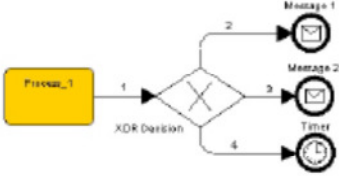
## Modeling the Sequence Flow of a Process

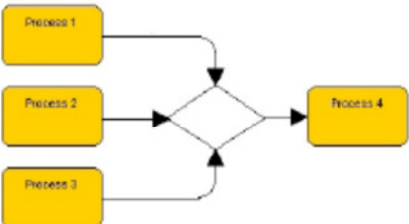

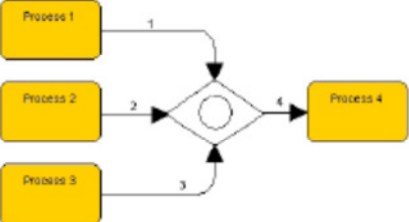
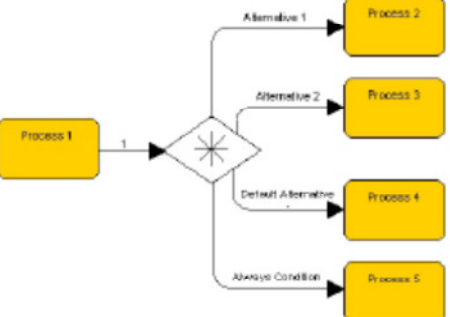
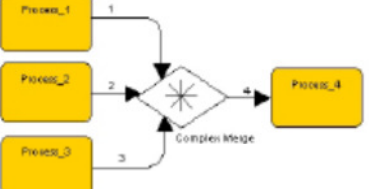
To show the order of execution of processes, you connect them with a Sequence Flow. A Sequence Flow is drawn as a line with a filled-in arrowhead (as shown in Figures 6 and 7). A Sequence Flow is used to show the sequence of processes in an organization or department. So if you have added pools or lanes to your diagram, you use Sequence Flow lines to connect events, processes, and gateways placed within the pools or lanes. BPMN makes a second flow line – the Message Flow – available to model ordering of processes between organizations or departments (in other words, between pools)[50].

## Modeling Decision Points with Gateways

Decisions, merges, forks, and joins in the process flow are modeled with a gateway symbol. A gateway can be thought of as a question that is asked at a point in the process flow. The question has a defined set of alternative answers, which are in effect gates. You may set

the stereotype of a gateway, and thus change the logic specified by it, and the symbol representing it, as described below.

Gateway Stereotype	Explanation
<p><b>Exclusive Decision (XOR)</b></p> <p>Data-Based XOR Decision:</p>  <p>Event-Based XOR Decision:</p> 	<p>XOR gateways are used to model data-based or event-based decisions. Data-based XOR decisions are the most common XOR gateways used. A data token traverses the Process Flow and arrives at the XOR gateway. The path that it flows out on is chosen based on condition expressions for each gate of the gateway. It can only go out on one flow.</p> <p>Event-based gateways are a recent development in business process management (BPM). An event-based XOR gateway represents a branching point where the alternatives are based on an event that occurs at that point in the process flow. A specific event, usually the receipt of a message, determines which of the paths will be taken.</p> <p>For example, you can model a process flow wherein the system waits for a response from a customer. The customer's response will either be a Yes message or a No message, and that determines which path is taken.</p>

Gateway Stereotype	Explanation
<p><b>Exclusive Merge (XOR)</b></p> 	<p>XOR gateways are used to model data-based or event-based merges. Exclusive means only one of many inputs is chosen to be output from the gate.</p>
<p><b>Inclusive OR Decision</b></p> 	<p>Inclusive (think 'including') means one or more of the outgoing Sequence Flows from the decision may be taken. There cannot be zero output flows -- you <b>must</b> specify a default flow.</p>
<p><b>Inclusive OR Merge</b></p> 	<p>Inclusive (think 'including') means that the process flow continues when the first input signal (a Token) arrives from any of the set of input Sequence Flows. If other signals subsequently arrive from the other input Sequence Flows, they are not used.</p>
<p><b>Complex Decision</b></p> 	<p>You specify a complex flow condition that references outgoing Sequence Flow names. The expression determines which output flow is taken.</p>
<p><b>Complex Merge</b></p> 	<p>You specify a complex flow condition which references incoming Sequence Flow names and/or process data that is coming into the gateway. The expression determines when the task starts.</p>

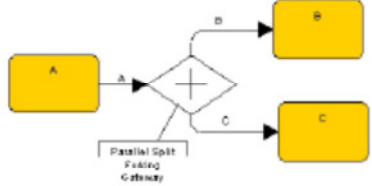
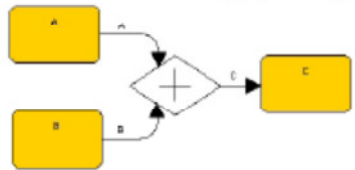
Gateway Stereotype	Explanation
<b>Parallel Forking (AND)</b> 	A Parallel gateway is also called an AND gateway. All Sequence Flows drawn out of the AND gateway are taken.
<b>Parallel Joining (AND)</b> 	The AND gate must receive an input signal (a Token) from <b>all</b> input Sequence Flows for the output flow to be taken. The process flow waits for all signals to arrive at the AND gateway before it can continue.

Table 3: Types of gateways and associated symbols

### Who Does What – Pools and Lanes

As you progress in modeling business flows, you take the processes, events, and gateways of the business process diagram and place them in pools or lanes. A pool is drawn as a rectangular region drawn horizontally across the diagram or vertically down it. A lane is a sub-partition within a pool and extends the entire length of the pool. Typically, a pool represents an organization, and a lane represents a department within that organization [58].

By taking processes and placing them in pools or lanes, you are specifying who does what, for events you specify where they occur, and for gateways you specify where decisions are made, or who makes them. The analogy between this representation and swimming pools is a useful one. You can imagine a process swimming down a lane, and changing lanes as need be to perform an activity, within a pool. The pool can be considered a ‘pool’ of resources. There are occasions when the process needs to jump to another pool, because it has different resources needed to complete the activity.

This is particularly apt where there is a need to describe B2B processes, where different organizations pass messages among one another to perform an activity [50].

A pool can represent other things besides an organization, such as a function (something that the organization performs, like Marketing or Sales or Training), an application (or computer software program), a location (a physical location in the company), a class (a software module in an object-oriented computer software program), or an entity (representing a logical table in a database). It can only represent one thing, but that thing comes from this ‘heterogeneous list’ of different types of things [50].

### **Modeling B2B Message Flows**

As mentioned previously, one goal of the BPMN business process diagram is to enable modeling of B2B messaging. To this end, the BPMN Business Process diagram offers the ability to model message flows. Traditional business process diagrams enable the modeling of sequential process flows -- from starting events to ending results. The BPMN business process diagram augments the Sequence Flow line with a Message Flow line, so that you can model people or machines sending messages to one another – an important part of depicting and understanding business-to-business and business-to-consumer processes [16, 58, 50]. BPMN specifies certain rules for modeling message flows and sequence flows. Sequence Flows can only be drawn among events, processes, and gateways within the same pool. Message Flows can only be drawn between events, processes, or gateways that exist in different pools – since messages are only passed between different organizations or applications, and so forth. BPMN suggests that these rules be enforced by the modeling tool providing BPMN support [16, 50].

There are times when you are modeling that you don't care how a process is performed in a company. It may be another company or a customer that is outside your scope; you have no control over it. You don't care how the company creates a message; you only care that the message has been delivered to you and contains information that you can use. Or you don't care what a company does with a message that you deliver to it – you trust that it does the right thing with it. You can treat the company (or application, function, and so forth) as a 'black box' – only draw Message Flows to or from the pool representing it, and not show any details inside the pool. This is in contrast to the pools that you model processes in, which can be considered 'white boxes' – you can see into them and examine their details [50].

### **Annotating the Models with Text**

A picture is worth a thousand words, so the saying goes. Conversely, sometimes a picture isn't enough – you need words to describe the nuances of something that a picture cannot do justice to. Thus BPMN provides you with a textual annotation that can be affixed to any model element, so that you may describe extra details about the element in good old-fashioned words. You may use Text Annotations on all model elements of the BPMN Business process diagram. Text Annotations are displayed within an open rectangle, attached to the symbol by a straight line [50].



### **3.5 Simulating business processes**

A model described using BPMN is a logical description of how the business operates, from which business process languages can be generated. However for optimal results this approach should be used hand-in-hand with business process simulation [50].

Simulation is a powerful technique available to business analysts to analyze their models prior to their realization. A model, when simulated, mimics the operations of the business, by stepping through the events in compressed time while displaying an animated picture of the flow. Because simulation software keeps track of statistics about model elements, performance metrics can be evaluated by analyzing the model output data. This enables you to avoid expensive mistakes by thoroughly reviewing the efficiency of a business model before actually implementing it [50].

### **3.6 How BPMN fits in with UML**

The advent of BPMN does not render obsolete the need for systems development, such as that performed using the Unified Modeling Language (UML) [56]. Systems development still has an important role to play in the overall enterprise architecture process. UML is a language that helps developers specify, visualize, and document models of software systems. It is very much targeted at system architects and software engineers. It has been developed as a means to streamline the software development process, from architecture design to application implementation for use by a technical audience. BPMN is targeted at business analysts, system architects, and software engineers. It has been developed as a way

to streamline the overall business lifecycle development process from process design – performed by a business audience [50, 56].

UML defines a number of diagrams that fall into one of three categories that describe:

1. Static application structure
2. Dynamic behavior
3. Management and organization of software solutions

Of these categories it is the dynamic behavior diagrams that are often used for modeling business processes, such as the UML Activity diagram and Use Case diagram. BPMN is related to UML in the sense that it defines a graphical notation for business processes that is similar to UML behavior diagrams. However, BPMN and UML have very different approaches to business process modeling [50].

UML offers an object-oriented approach to the modeling of applications, while BPMN takes a process-centric approach. Most UML methods ask you to find the objects first using static structure diagrams, and then ask you to build dynamic behavior diagrams to show how objects interact. As a way to model, this method is alien to most business analysts [50, 56]. BPMN offers a process-centric approach that is more natural and intuitive for the business analyst to use. With BPMN, control and message flows of processes are modeled first. An object model for the process is defined implicitly rather than explicitly. BPMN also offers you the option of explicitly modeling business objects that may be exposed through business services in your process flows [58, 50].

UML is an assemblage of diagrams that are the results of the collective best practices of the various founding practitioners. Unfortunately, what this means is that the diagrams are an aggregation that have not been specifically designed to work with each other. As a

consequence, developers can only model part of their applications with UML; the detailed implementation level is not covered. In contrast, BPMN defines a single type of diagram that has multiple views derived from the same underlying process execution meta-model. The natural result of this is that implementation in a business process execution language merely becomes another logical view of the process [50].

### **3.7 Example of process creation in BPMN**

In this section will be described through an example how is possible to create a model in BPMN for a particular business process. The aim is to make BPMN more clear and simple to understand.

Let's take into consideration a Credit Application process. A Credit Application process begins with the recording of the application where the client expresses an interest in acquiring credit. This stage includes the presentation of the application, and the required documents to the organization for verification. This is followed by an analysis or study of the credit application and finally we find the activities needed to either disburse the credit or to notify the client in case of rejection [15]. The respective process modeled in BPMN is presented in Figure 8.

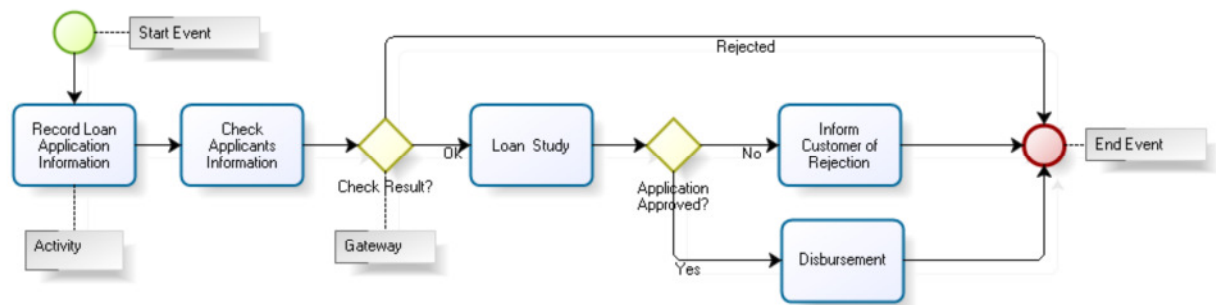


Figure 8: BPMN model of a credit application process

As presented in the above example, in a Business Process Diagram there are a number of graphical elements with which we represent a business process. In the above example we can see different types of elements that describe how the process works. Within these elements are the activities that represent the work that was carried out, the beginning and end events, which indicate the starting point and completion of the process, plus the decision elements known in BPMN as Gateways, which indicate alternatives along the way. These elements are connected by means of Sequence Lines that show the process flow. At the beginning of the Credit Application Process there is the figure “Start Event”, which indicates the beginning of the process. Processes can begin in different ways and BPMN provides for different types of Start Events (simple, message, signal, etc.) [58, 50, 15]. At the end of this process we find the figure “End event”, indicating termination of the process. As the graph shows, the process ends when the applicant is rejected, the credit application is not approved or the loan is granted and disbursed [15].

The gateway used in the above example is the Exclusive Gateway. As a decision element, this gateway behaves like an “XOR”, in other words, only one of several given alternatives can be taken. In the Credit Application Process we can see two examples of the use of an

exclusive gateway. The first one depends on the result of verifying the applicant's information: the line may run in one of two directions; if the result was "Applicant Rejected", the process ends there, and if the applicant was accepted, the process can continue. In the second example, the decision is based on the result of the credit study: if the application is rejected, the client is notified, if it is approved the credit is disbursed [58, 15].

If we look into the Credit Application Process, we discover activities that can be analyzed in greater detail. One of these activities is checking the Information provided by the applicant. Credit organizations normally carry out several analyses of an applicant, verifying, for instance, if the applicant is already a client of the organization, if they are a target client or, check the applicant's financial situation [15]. Depending on the above, the activities may or may not be compound. As we already described in the previous sections in BPMN, compound activities are known as Sub processes, and atomic activities as tasks. The process flow diagram of a Credit Application would look like the one below, when the Information Checking activity is included as a sub process.

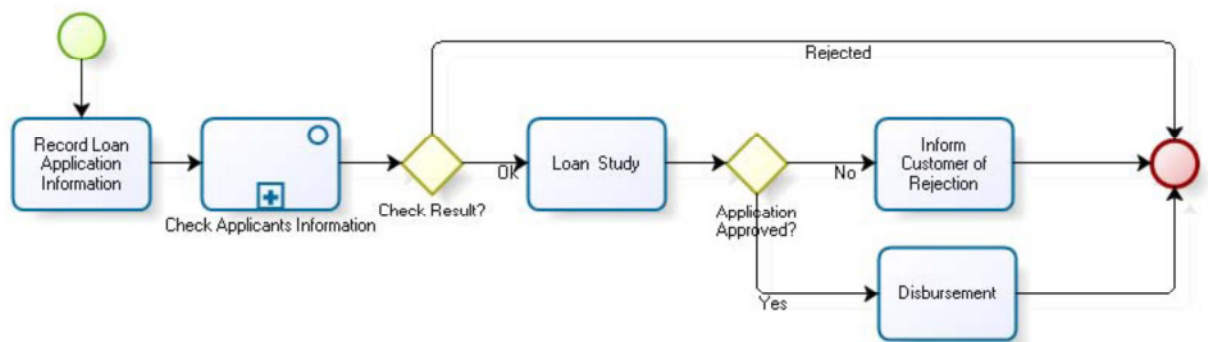


Figure 9: Credit application extended process

The sub process of verifying the applicant's Information would be as per below.

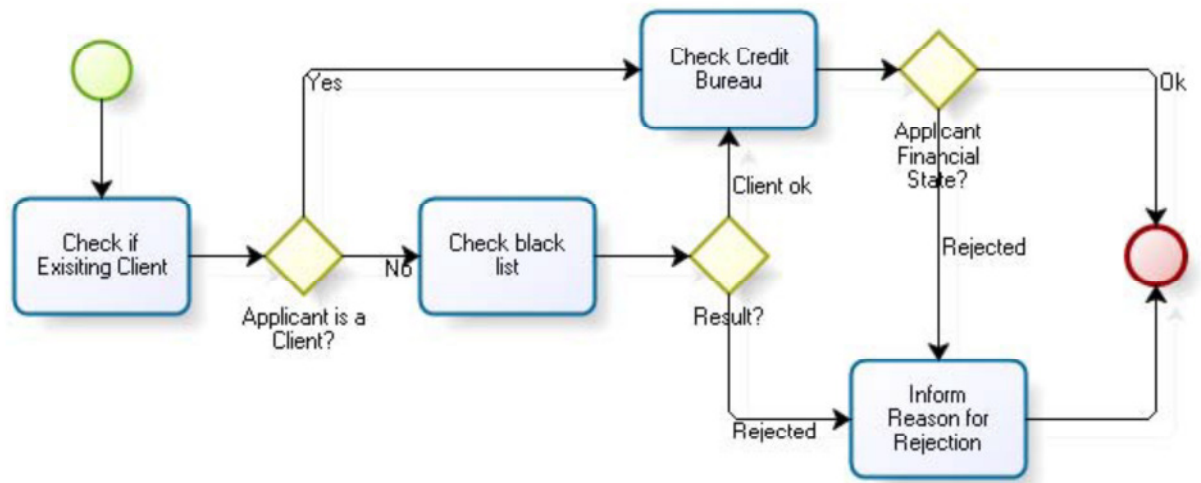


Figure 10: Verifying the applicant's information sub process

Additionally, within the Client Information Checking Sub process, we find that the activities of Verifying for Existing Client, Checking the Client Black List and Credit Bureau Consultation are automatic tasks, that is to say, they are carried out by a system with no human intervention. This can be either an automatic device or a Web service. For diagramming these elements, BPMN provides a type of task called Automatic Task (Service) [15]. The sub process of Verifying Applicants' Information is shown in the following way, with the Automatic activities:

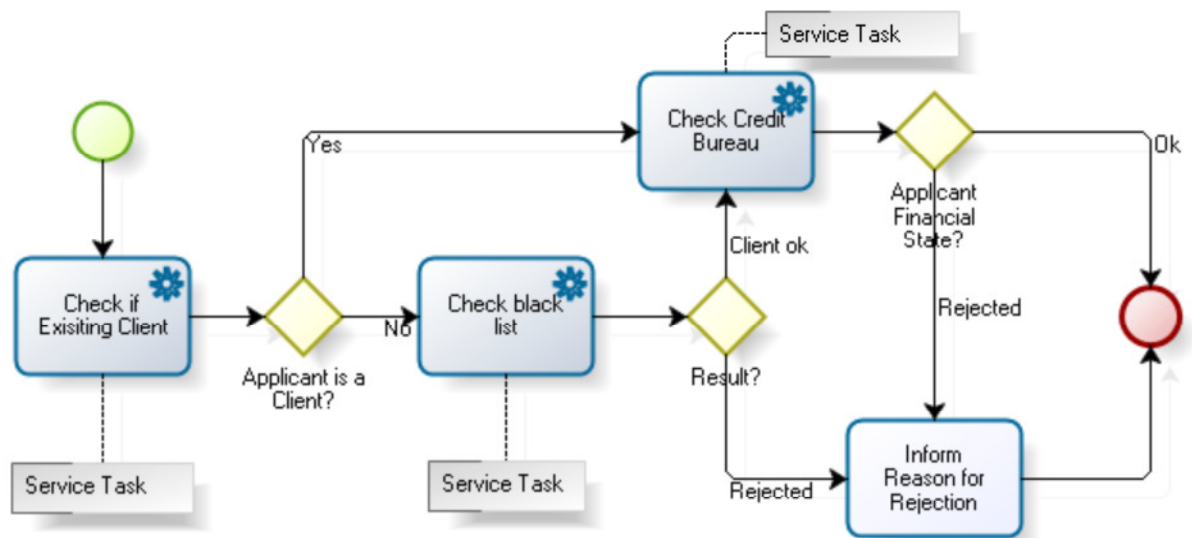


Figure 11: Verifying the applicant's information sub process updated with Automated Tasks.

## **CHAPTER IV: MODEL DRIVEN ARCHITECTURE**

### **4.1 What is the motivation?**

Currently all the evidence shows that the development of software products (commonly referred to as applications) has always been and still is a complex task, especially in management environments (which are linked to Organizational Systems). What is worse is that their complexity, rather than decreasing, continues to increase. This has become a constant factor in information systems development and is due mainly to the fact that the customers' needs for products are always more sophisticated. Also, the resulting applications are developed using highly dynamic, constantly changing technologies, which are usually presented with a facade of simplicity [53].

Complexity also increases because development technologies must structure the final software product according to protocols that are associated to different architectures, which are usually based on object-oriented models and on the development of distributed components and their associated software architectures (DCOM, CORBA, multi-tiered architectures as an evolution of the traditional client/server architectures, internet/intranet environments, etc.) [53].

In most of the cases, the technological complexity leads with the software engineer devoting more effort to getting to know the technical aspects of a particular solution, rather than focusing on understanding the business problem to be solved by the application. This approach raises the following questions: modelling vs. programming; understanding and representing the problem vs. implementing the solution; focusing on the modelling (the



problem space) and temporarily ignoring the technical aspects (the solution space) [52, 7].

It is clear and we should always remember that only after we have a thorough understanding of what the software product must do, will we be able to define how it should be built. It is absolutely necessary to have production methods that allow specifying and representing the Conceptual Schema of an Information System and to then transition to the implementation phase, which is responsible for producing a software product that is functionally equivalent to the specification created in the conceptual modelling phase [45, 7].

Perhaps the answer to the question why methods like the above one desired are not common, is simply that we do not possess methods and development environments that are advanced enough to allow the construction of software applications from conceptual models, methods that are advanced enough to allow the engineer to focus on what the system is, and not on how the system will be represented in a given programming environment, methods that are advanced enough to allow the use of notations that are close to the problem space, and not the solution space [53].

During the past years there is clear evidence how the level of abstraction in programming languages, tools and environments has evolved, coming ever closer to the user space. This is only natural, if we take into account that any analysis of an information system is characterized by the existence of two fundamental actors [53]:

1. The stakeholders, who know the problem to be automated by a software product.  
They are the final users of this software product.
2. The application developers, who must build a coherent model from the knowledge of the stakeholders. The resulting Conceptual Schema must represent (in the

Problem Space) the concepts comprehended and captured during the interaction with the stakeholders. Naturally, the next step is to correctly represent this Conceptual Schema by using the specific set of software representations provided by a given programming environment, in what we have denoted the Solution Space [53].

From the times of the first assembler languages to the modern conceptual modelling environments and the production of software products driven by conceptual models, Software Engineering has tried to provide application developers with production methods that are closer to the ProblemSpace with the twofold goal of:

1. Easing the specification of Information Systems to make it feasible in terms that are close to the knowledge of the stakeholders.
2. Easing the conversion to the notation of the Solution Space in an organized and precise manner by defining the corresponding transformation mechanisms (the ideal case being the use of model compilation techniques) [52, 7].

As a result of this, one of the critical tasks in modern application development and process automation is Requirements Specification, and the Conceptual Modelling tasks associated with it that lead to building a Conceptual Schema where these requirements are adequately represented. This phase of Conceptual Modelling is strongly related to the final quality of the software product and to the productivity of the corresponding software process because the requirements describe the goals of the effort to develop the software, provide guidelines for the design of the software architecture, and set the basis for measuring the quality of the final product [53].

The most critical tasks in the software production process are still the specification and analysis of requirements. It is also recognized that errors produced in this phase of the software development process could have a huge impact on the reliability, cost and robustness of the system [20]. These errors are commonly attributed to the lack of tools that offer integral support to the development process and that are closely linked to the subsequent phases of the development process. An immediate conclusion is that 21st century application developers require new development tools that provide them with higher-level constructs, so that they can specify applications using concepts that are closer to the ones used by humans in our cognitive and communicative processes. The “programs” that are built with these conceptual primitives or conceptual patterns of a higher level should then be transformed into the equivalent software product through a translation process that associates each conceptual primitive to its corresponding software representation. Under this hypothesis, the automatic generation of programs from Conceptual Schemas is no longer an unreachable dream but a solid reality [53].

It is also useful to analyze the development methods that are currently being used in the software industry. This analysis shows that these methods propose development processes that consider software construction to be the completion of a high number of tasks in several phases. In most of the cases, the resulting process is extremely complex and of little use in day-to-day practice. Another commonly observed issue in current approaches is that some of the effort put into the completion of some of the tasks and in the production of documentation has little or no effect at all on the final product. That is, in the line of what we could refer to as “traditional CASE” methods, much of the effort required to set up the model of a system is often nothing more than an elegant (in the best of cases)

documentation of it, but this then has to be manually transformed into a software product by using a notation and concepts that are totally different from those used to build the model [45, 7].

This “semantic gap” between the model notation and the programming language usually makes the use of a CASE tool a problem because engineers not only have to obtain the software product but they have to model it as well. This explains the problems that have historically prevented the universal use of CASE. In addition to this, when maintenance problems arise and there are changes to the specification of the systems, it is almost impossible to avoid the temptation of performing the modifications directly to the software product, so that the model and its implementation usually are not synchronized. To avoid this situation, the construction of information systems of a certain size and complexity requires the use of methods to carry out the development process in a rigorous and systematic way. These methods must perform the phases that are strictly needed to obtain a quality software product in a practical and productive manner. Experience shows that in Software Engineering, as in any other discipline where the goal is to obtain a certain quality product, simplicity and effectiveness are the two quality attributes that must be co-guaranteed. Software production methods that are based on an exaggerated and unjustified number of tasks and phases will simply not be used in day-to-day practice and will inevitably be discarded by developers [53].

#### **4.2 Current information systems development process**

Applications that are associated with an Organizational Information System usually share the following common features [53]:

- Huge volumes of data are stored in databases that act as repositories of persistent information and represent the structured data architecture of the system under analysis;
- Great amounts of lines of code are programmed to build user interfaces to provide the different users of the resulting application with different ways of accessing data and services;
- Every user must be provided with the correct implementation of the functionality available, so that the business rules are adequately represented and there is a customized functional appearance;
- The aforementioned levels of presentation (or user interface), functional and persistence (or database) must be conveniently interrelated so that it is possible to: set up the execution of a service by collecting the values of all arguments needed to enable execution; send the appropriate message with these data from the presentation layer to the functional layer; and retrieve the data involved in the execution of the service from the persistence layer and store the data result of this execution;
- A large number of users must be able to use the application from different environments, which are usually distributed over a network [53].

To incorporate these features, the construction of this type of software products requires technical knowledge about matter that differ as strongly as database design and

construction, conceptual modelling, user-interface design, middleware technologies, programming languages, operating systems, prototyping tools, test tools, etc – and these are only the elementary ones. With such a volume of knowledge and technology required, and with the need for the integration of all these disciplines so that the resulting software product is correct, the greatest hurdle is the path to successfully delivering a new project is the ability to integrate all the tools involved into a single development environment and to incorporate all the knowledge required to implement these huge applications in a single team [53].

We are witnessing a continuous evolution in the context of application development due to the fact that the required levels of software product quality are becoming ever higher for both the applied technologies and the stakeholders' requirements. However, the root cause of the problem of today's development of large software products is that technology is constantly changing and becoming more complex, without helping to increase the quality of the software process nor the software product. All of this is happening in a context in which the dominant factor is the lack of adequate tools to hide all of this technological complexity from the developer, and thereby ease the process of obtaining the complete and correct software product that is desired [52, 53].

The current modern development tools share two generic problems:

- First they do not provide developers with high-level constructs in a well-defined methodological framework. Therefore, applications are typically constructed from code fragments that are at a level lower than that of the problem specification. The semantic gap between the Problem Space and the Solution Space, between modelling and programming, usually causes the system specification that is created

in the scope of the Problem Space to be incorrectly represented in the scope of the Solution Space. This happens because the constructors that are common in the programming environments are not suited to represent the real-world models produced in the conceptual modelling phase in a direct and easy way. Additionally, the existence of two different processes and notations (which must be kept consistent) does nothing but introduce an unavoidable level of complexity and noise into the software production process [53].

- Secondly they do not adequately hide the technological complexity, and often do exactly the opposite: technology and its inherent complexity are sometimes promoted as a feature of the development environment, rather than being kept as low-level implementation details. Most market tools are too specialized and focused on only some part of the development process, so that many of these have to be integrated with others in order to have a cohesive development environment that covers all the phases in the development of a software product. However, this too has a clear impact on the ever-increasing complexity of the development environment [53].

Applications development with current development tools becomes an intensive programming task, which is always working at an abstraction level that is too low and, in the context of the SolutionSpace: moving data in the database, manipulating the user interface to capture user actions that are then sent to the functional part of the application, etc. Most of the produced code is tightly coupled to the underlying technology: it necessarily relies on operating systems, databases, middleware products, etc [53].

Current development processes are focused on the codification (the “how”, or the Solution Space), rather than on modelling (the “what”, or the Problem Space), which negatively affects productivity and the quality of the final software product. Curiously enough, this situation does not occur in other engineering guilds: no engineer ever thinks about beginning the construction of anything without having the “blueprints” where the stakeholders and technical requirements are clearly specified. Needless to say, these blueprints continuously guide the development process of the subject under construction (the equivalent in Software Engineering would be the software product). The current situation brings about some remarkable negative effects, as discussed below.

- The analysis and design documents produced to specify the problem and the application architecture are seldom used in a straightforward manner to obtain the implementation. In the best cases, they are used to specify the basic system architecture; however, from that point on, a huge “manual” programming effort is required to convert these models into a finished product. Most of the time, these system models are used as mere documentation that becomes obsolete as the products evolve, so that the effort put into creating the models is useless.
- Communication problems arise between developers because of the differences in the process and notation used in the Problem Space (where analysts and modelers work) and the Solution Space (where designers and programmers work). In organizations where different groups coexist and cooperate to carry out each of the development tasks, the chances are high that errors will appear and communication



problems will occur. This results in long iterations in the specification-design-implementation-test lifecycle.

- Due to the ever-increasing complexity of technology, large development teams that consist of members with great amounts of technical knowledge and great programming skills are needed. As a consequence of technological complexity, large team sizes create communication problems and bureaucratic overload that negatively affect productivity.
- It takes a lot of time and effort to obtain a prototype that is functionally equivalent to the system requirements specification and that can be delivered to the user, and so user feedback usually takes place at later stages of the lifecycle. This delay in feedback raises the cost of introducing changes because the product will be in an advanced state of implementation when the changes are requested [53].

#### **4.3 Better development environments are needed**

The issue is that producing an Information System today is costly (because expensive resources have to be used over extended periods of time), much too slow for modern business conditions, very risky (in the sense that it is hard to control and has a high failure rate), and highly unsafe (because it introduces hidden failure points) [53].

The main problem is that – from a high-level, software process-oriented perspective – the development process has not changed much over the past 40 years; that is, the task of programming is still the “essential” task. Programming is the key activity associated with the fact of creating a software product. It cannot be claimed that programming technologies

have not improved year after year, with new proposals and their associated tools. However, the issue here is that, considering the very bad results the programming-oriented techniques are historically obtaining, a simple question arises: is it worth searching for a better way of producing software [53]?

We should ask ourselves why so many software systems historically fail to meet the needs of their customers. For decades, the “silver bullet” has apparently been provided to the community in the form of some new, advanced software technology, but always unsuccessfully. We have a large list of technologies that were intended to solve the problem: we could set down in this list Assembler, Third-Generation Languages, Relational Databases, Declarative Programming, Methodologies and CASE tools (Structured Analysis and Design initially, Object-Oriented extensions for Analysis and Design, UML-based, subsequently), Component-based Programming, Aspect-based, Agent-Oriented, Extreme Programming, Agile Methods, Requirements Engineering, Organizational Modelling, etc. Nevertheless, the same “phantom of the opera” is always present: the unsolved Crisis of Software notion [53].

What we now need are new techniques, methods and development environments that enable us to take the software development process to a higher abstraction level by using an expressiveness that is much closer to the Problem Space, thereby increasing the productivity and quality of the application development process. Current approaches for this goal are modelling tools with code generation capabilities, or complex prototyping tools that attempt to help developers by providing them with a way of rapidly iterating the production process. The common problem is that applications produced by such tools are

used only once and then discarded, and so the effort put into the construction of the prototype seldom pays off [53].

Another remarkable fact is that the term “code generation” has traditionally been misused. Most modelling environments use the “code generation” tag as a marketing claim. Nevertheless, the term is used in a deceptive way because it never applies to tools that produce code from conceptual schemas, as a conceptual schema compiler would do. At best, such tools partially transform the conceptual specification of the system into general-purpose code templates, which are often of little or no use from the point of view of the final software product. If these templates are to be reused, something that is feasible from the architectural point of view of the final real product, then they must be adapted in a nontrivial way, and manually completed according to the technological features of the development environment in use. This adds to the problem of the unavoidable desynchronization (lack of synchronization) between the conceptual schema and the resulting software product, when the modifications and changes are applied to the latter but not to the former [53].

Having said this, we can now introduce the desirable features that should exist in the advanced, “ideal” development environment.

1. It must provide us with facilities to rapidly construct quality applications, quality being the functional equivalence between user requirements, the conceptual schema, and the resulting software product.
2. It must also allow team development, and cover the entire software production lifecycle in an iterative and incremental way. To do so, it must provide a clearly specified and structured software production process.

3. It must effectively hide all the technical complexity, focus on the specification of the problem to be developed, and provide a complete and correct set of conceptual primitives to act as basic constructors in the elaboration of a correct and complete conceptual schema.
4. Such basic constructors (conceptual patterns or conceptual primitives) must have an underlying formal support to ensure syntactic and semantic consistency. An interesting alternative is the use of information system formal specification languages as a high-level data repository. This hides the complexity that is historically associated to such languages from the developer, and allows the introduction of relevant information by means of graphical editors, using graphical notations that are compliant with widely used standards such as UML [56]. Thus, the best features of formal and conventional modelling techniques can be combined in a practical and effective way.
5. All of this must also allow the specification of a system regardless of the technological and programming aspects. The work of software engineers should be focused on the scope of the Problem Space, where the main goal is to specify the “what”, regardless of the “how” that is typically associated with the low-level considerations of a concrete representation in the Solution Space, and with the inherent complexity of any software technology [53].

The best way to ensure this functionality is to allow the real, automatic generation of code for the full application from models built by analysts. The term “real” is important in this context. The goal is that the Conceptual Schema can be compiled directly by applying a transformation mechanism based on the definition, design and implementation of a finite

set of mappings between conceptual patterns and their associated software representations. This will constitute the core of the model compiler that, in turn, will be the core of this new breed of development environments [53].

#### **4.4 Model Driven Architecture**

Computing infrastructures are expanding their reach in every dimension. New platforms and applications must interoperate with legacy systems. Virtual enterprises span multiple companies. The Internet is imposing new integration challenges as it extends into every corner of every organization. New implementation platforms are continually coming down the road, each claiming to be "the next big thing." Those who architect computer systems, whether for banks or battleships, face daunting technology choices. To protect their investments and maximize flexibility, they buy hardware that implements open interconnection standards like Ethernet and USB, and software that uses open interface standards like CORBA. It's the only sensible course in today's rapidly changing, multi-vendor computing environment [30].

But as computers and networks become faster and cheaper, even interconnection standards must evolve. New technologies constantly appear for new application niches. One need look no further than the recent rise of XML to see how quickly this can happen. How can organizations ensure that their mission-critical information systems are rooted in standards that will adapt to new hardware capabilities and software platforms [30]?

OMG addresses this reality with MDA, the Model Driven Architecture. MDA supports evolving standards in application domains as diverse as enterprise resource planning, air traffic control and human genome research; standards that are tailored to the needs of these

diverse organizations, yet need to survive changes in technology and the proliferation of different kinds of middleware. The MDA addresses the complete life cycle of designing, deploying, integrating, and managing applications as well as data using open standards. MDA-based standards enable organizations to integrate whatever they already have in place with whatever they build today...and whatever they build tomorrow [30].

MDA provides an approach for deriving value from models and architecture in support of the full life cycle of physical, organizational and Information Technology systems. The MDA approach represents and supports everything from requirements to business modeling to technology implementations. By using MDA models, we are able to better deal with the complexity of large systems and the interaction and collaboration between organizations, people, hardware, software [43].

The primary feature of MDA which enables us to deal with complexity and derive value from models and modeling is defining the structure, semantics, and notations of models using industry standards – models conforming to these standards are “MDA Models”.

MDA models can then be used for the production of documentation, acquisition specifications, system specifications, technology artifacts (e.g. “source code”) and executable systems. MDA leverages models to enhance the agility of planning, design, and other lifecycle processes, and improve the quality and maintainability of the resulting products. MDA can also be used to relate the models to implemented solutions and related artifacts, such as when transforming software design model to executable code, or relating the design model to its requirements. The MDA set of standards includes the representation and exchange of models in a variety of modeling languages, the transformation of models, the production of stakeholder documentation, and the execution of models. MDA models

can represent systems at any level of abstraction or from different viewpoints, ranging from enterprise architectures to technology implementations. MDA “connects the dots” between these different viewpoints and abstractions [45, 43].

#### **4.5 The MDA approach to deriving value from models**

The essential goal of MDA is to derive value from models and modeling that helps us deal with the complexity and interdependence of complex systems. There are multiple ways to derive value as summarized below [43].

##### **Models as communications vehicles**

A fundamental value proposition for models and modeling is to facilitate a team or community coming to a common understanding and/or consensus. MDA standards assist in three ways [43]:

- Providing well defined terms, icons and notations that assist in a common understanding of a subject area
- Providing the foundation for models as semantic data to be managed, versioned and shared
- Providing libraries of reusable (asset) models such as common vocabularies and rules, reusable processes, business object models, or architectural design patterns

Models also become part of the “corporate memory” of an organization, capturing the requirements, design and intent of its organization, processes, information and services [43].

##### **Derivation via automated transformation**

Deriving artifacts and implementations from models may be fully or partially automated. Automation reduces the time and cost of realizing a design, reduces the time and cost for changes and maintenance and produces results that ensure consistency across all of the derived artifacts. For example, manually producing all of the web service artifacts required to implement a set of processes and services for an organization is difficult and error-prone. Producing execution artifacts from a model is more reliable and faster. Producing derivative artifacts can also be intermixed with model execution, simulation and analytics [43].

Automated derivation involves a platform independent “source model” with some parameters for how that source model is to be interpreted, a “transformation” and a platform specific “target artifact”. In many cases the target artifact is another model. Derivation also includes logical inference as defined in ontology languages such as OWL and common logic. Producing “code” from models is one of the first uses of MDA and a continued value proposition. However, as we will see below, it is not the only value proposition [43].

### **Model Analytics**

Once models are captured as semantic data various analytics can be executed across those models including model validation, statistics and metrics. Analytics assists in decision making, monitoring and quality assessment [43].

### **Model Simulation and Execution**

Models as data can drive simulation engines that can assist in both analytics and execution of the designs captured in models. Simulation assists in the human understanding of how a modeled system will function as well as a way to validate that models are correct. Models



can, in some cases be directly executed – serving as the “source code” for high-level applications that implement processes, data repositories and service endpoints. Model execution provides a direct and immediate path to realizing a design with a minimum of technical details being exposed. DBMS Schema and process models are well-known categories of (partially) executable models [43].

### **Deriving information from models**

Well defined models can be used to derive other information – information that can be inferred directly from information in source models as well as models and other artifacts.

Derivative information that can be derived from models includes [43]:

- Process playbooks
- Derived insights

For example from a set of process models, a tool could derive responsibilities of a particular organization unit represented on one or more swimlanes, or discover an unanticipated side-effect.

- Documentation
- Acquisition specifications (i.e. RFPs)
- I.T. Systems

### **Structuring Unstructured Information**

There are vast quantities of unstructured data in documents, web pages and diagrams.

While the MDA technologies do not directly address extracting unstructured information, MDA does provide a way to define the structured representation of these documents.

Software that scans or extracts information from unstructured text can use MDA models as the repository for the resulting manually enriched semantic data [43].

#### **4.6 The structure and semantics of models**

The foundation of MDA is capturing models as data that we can query, analyze, report on, validate, simulate, and transform into other useful formats (such as executable code). There are MDA standards for how we represent models as data, standards for the meaning of model data for a multitude of purposes, standards for how we render model data and standards for transforming between different representations for different purposes [43]. Consider a “whiteboard” session with a team of collaborators. In that session you may draw some diagrams on a whiteboard that help communicate concepts to achieve a mutual understanding. That mutual understanding comes from the team having a common understanding of the symbols used on the whiteboard, perhaps the presenter explained the symbols they used, or perhaps these are understood within your community. This whiteboard diagram can have a lot of value within that meeting; it is a useful informal model. But, after the meeting it is lost; perhaps we take a picture of the white board so we can keep a copy, or perhaps we reproduce it in a drawing tool. Another example of an informal model is the design every software developer has in their head prior to writing code. But such white-board sessions and “in the head designs” have their limits – we need something more durable [43].

What we need is data – data with well-defined structures and meanings that represents the facts and concepts behind the pictures. The data is “behind” the diagrams; in fact diagrams can be thought of as one possible rendering of data for a particular purpose. The

whiteboard session is great for a meeting but as soon as possible we want to capture the information represented on the white board as data with well-defined meaning. The well-defined meaning is the semantics of the data, the definition of what the data means in the context of our models [43].

What can we do with models as semantic data? Once we have the data semantics behind the diagrams we can [45, 43]:

- Manage and evolve the data
- Query, validate and analyze the data
- Share the data with others
- Repurpose it for other uses
- Make connections between related models
- Make other diagrams or renderings that may make sense to others
- Derive derivative value using software tools

### **What We Model – the Domain Subject Area**

There are models of different things for different purposes. The subject area establishes the context and scope of the model. Each subject area is considered a domain. High-level domains include [43]:

- Telecommunications
- Healthcare
- Retail
- Manufacturing
- Transportation

- Defense
- Government
- And many others...

A more specific domain may be the subject of a particular model for a company, government agency or defense mission. These are models of the system we design and want to make real. Different perspectives on the same subject area are considered viewpoints. These different viewpoints may look quite different so as to be meaningful to different stakeholders. The models can be segregated based on the domain and viewpoint these domains and viewpoints [43].

#### **4.7 Basic concepts of MDA**

##### **System**

A system is a collection of parts and relationships among these parts that may be organized to accomplish some purpose. In MDA, the term 'system' can refer to an information processing system but it is also applied more generally. Thus a system may include anything: a system of hardware, software, and people, an enterprise, a federation of enterprises, a business process, some combination of parts of different systems, a federation of systems - each under separate control, a program in a computer, a system of programs, a single computer, a system of computers, a computer or system of computers embedded in some machine, etc. One of the key strengths of modeling, and one that distinguishes it from implementation technologies like software source code, is that it is an excellent way to represent, understand and specify systems [43].

## **Model**

A model in the context of MDA is information selectively representing some aspect of a system based on a specific set of concerns. The model is related to the system by an explicit or implicit mapping. A model should include the set of information about a system that is within scope, the integrity rules that apply to that system, and the meaning of terms used [43].

A model may represent the business, domain, software, hardware, environment, and other domain-specific aspects of a system. Such a model can include many kinds of expression; for example a model of a software system could include a UML class diagram [56], E/R (Entity-Relationship) diagrams, and images of the user interface, while a model of a physical system could include a representation of the hardware and physical environment, and a performance simulation. A model of an enterprise may include business processes, services, information, or resources. Some of these expressions are human readable; others exist in electronic media. For these expressions, there may exist several notations and formats. By a single model, we mean the whole set of expressions constituting information about a system [43].

## **Modeling Language**

To be useful, any model needs to be expressed in a way that communicates information about a system among involved stakeholders that can be correctly interpreted by the stakeholders and supporting technologies. This requires that the model be expressed in a language understood by these stakeholders and their supporting technologies. Achieving this understanding implies that the structure, terms, notations, syntax, semantics, and

integrity rules of the information in the model are well defined and consistently represented. The structure, terms, notations, syntax, semantics, and integrity rules that are used to express a model constitute the modeling language. Well-known modeling languages include UML, SQL Schema, BPMN, E/R, OWL, and XML Schema [43].

In that we have models and modeling languages there must be some connection between them: a model is said to conform to a modeling language. That is, everything that is said in some model is allowed to be said by the modeling language. Formal modeling languages have some way to determine if a model conforms to a modeling language. In contrast, informal modeling languages (perhaps one used on a white board) have no such test of conformance – their meaning is then more easily misinterpreted or lost [43].

There is a certain circularity to models and modeling languages. The best way to express a modeling language is as a model – this is called a metamodel. A metamodel is a model that defines a modeling language and is also expressed using a modeling language. Modeling tools typically help a modeler create models that conform to the metamodel of the language they are using [43].

## **Architecture**

The purpose of architecture is to define or improve systems or systems of systems. The architectural process encompasses understanding the scope of the systems of interest, understanding stakeholder requirements, and arriving at a design to satisfy those requirements. The two word-senses in which architecture is used are [43]:

- A set of models with the purpose of representing a system of interest.

- The activity and or practice of creating the set of models representing a system.

Model Driven Architecture advocates the application of modeling to the architectural process and formalizes the resulting artifacts such that the realization or improvement of the system may be more actionable, less expensive and less risky [43].

### **View and Viewpoint**

A *viewpoint* specifies a reusable set of criteria for the construction, selection, and presentation of a portion of the information about a system, addressing particular stakeholder concerns. A *view* is a representation of a particular system that conforms to a viewpoint. We could, for example, have a view representing the security concerns of a payroll system. In addition, if we are creating a model of a system that shares information there may be views for [43]:

- The meaning of the information to be shared
- How the information is to be structured when shared
- What information is to be shared with whom and under what conditions
- The protocol by which the information consumer requests or is sent the information
- Integrity and business rules
- What technology is used to actually transmit the information
- How the shared information is derived from our internal data

Note that the above views are interrelated, that is there are connections between them that may or may not be visible in any one viewpoint [43].

## **Abstraction**

Abstraction deals with the concepts of understanding a system in a more general way; said in more operational terms, with abstraction one eliminates certain elements from the defined scope; this may result in introducing a higher level viewpoint at the expense of removing detail. A model is considered more abstract if it encompasses a broader set of systems and less abstract if it is more specific to a single system or restricted set of systems. For example a very abstract concept of a postal address may be a piece of information that allows for the delivery of mail to a specific location. A less abstract definition may enumerate qualities of a postal address, such as name, city, country, state and postal code. An even less abstract form may be a SQL Schema for an address while a fully non-abstract postal address could be *8605 Westwood Center Drive, Vienna VA USA* [43].

Modeling and abstraction go hand-in-hand and allow us to understand and even specify a system while “abstracting away” some details such as the specifics of how it may be implemented in a particular platform or technology. The more abstract a model the wider array of systems it represents, but a more abstract model also needs more details filled in for it to be able to be actionable [43].

Another important dimension in abstraction is the relationship of a model element to the thing it is modeling: some model elements (e.g. a class "customer") represent a pure conceptual thing, whereas other model elements (e.g. the instance specification "Barak Obama") represent things that really exist in the real world. One of the capabilities of MDA is automating the transformation between levels of abstraction by the use of patterns. For example, given a more abstract definition of a postal address we may be able to apply a pattern to produce a SQL Schema representation for an address. This capability is the



foundation of the “technology independence” MDA is known for and the automation of software solutions from models [43].

### **Architectural Layers**

It is useful to identify particular “layers” of an architecture with respect to its level of abstraction. While there can be any number of architectural layers, a broad categorization of this concept is [45, 43]:

- Business or domain models – models of the actual people, places, things, and laws of a domain. The “instances” of these models are “real things”, not representations of those things in an information system. In MDA domain models have historically been called a “CIM” for “Computation Independent Model” [52].
- Logical system models – models of the way the components of a system interact with each other, with people and with organizations to assist an organization or community in achieving its goals.
- Implementation models – the way in which a particular system or subsystem is implemented such that it carries out its functions. Implementation models are typically tied to a particular implementation technology or platform [43].

### **Transformation**

Transformation deals with producing different models, viewpoints, or artifacts from a model based on a transformation pattern. In general, transformation can be used to produce

one representation from another, or to cross levels of abstraction or architectural layers [52, 43].

When transformation is used to produce one representation from another, the conceptual content stays the same but the information is presented in another form. For example, you may create HTML documentation for an information model – but the information model content is the same – just available in HTML. Often, transformations combine multiple input models into an output model. For example, a technology-independent model may be transformed into a technology-specific model by combining it with a model of the technology to be used (e.g. J2EE) [43].

In that you can have models of different levels of abstraction and different concerns that represent different yet related viewpoints, transformation can be used to cross levels of abstraction, going from a more abstract representation and, based on patterns, producing a more concrete representation – this is essentially “forward engineering” with MDA. It is also possible to go from a concrete representation (Say, a XML Schema) to a more abstract representation – say a UML class model. MDA transformation specifications provide the mechanisms to transform between representations and levels of abstraction or architectural layers [43].

### **Separation of Concerns**

One of the essential characteristics of quality architecture is separation of concerns. When concerns are separated it is possible to deal with, understand and specify one aspect of a system without undue dependencies on other aspects. Separation of concerns enables greater agility, ability to deal with change and a “divide and conquer” approach to realizing

a system. MDA helps support separation of concerns by enabling different viewpoints of a system as well as the transformation between levels of abstraction. In particular, MDA focuses on the separation of the business concerns of a system from the technology-dependent implementation concerns of components of that system. This enables technology independence, agility and resilience in a dynamic environment where both business and technology concerns may change over time or across organizational boundaries [43].

## **Platform**

A platform is the set of resources on which a system is realized. This set of resources is used to implement or support the system. In the context of a technology implementation, the platform supports the execution of the application. Together the application and the platform constitute the system. The application provides the functional part of the system as described by the model. Here are examples of types of platforms [43]:

### *Business or domain platform types*

- An organizational structure
- A set of buildings, machines or other physical plants
- Employees of an organization (human resources)
- Agreements supporting an organization or community

### *Computer Hardware and Software platform types*

- Batch: A platform that supports a series of independent programs that each run to completion before the next starts.

- Dataflow: A platform that supports a continuous flow of data between software parts.
- Embedded: A platform specialized for detection of changes in and exercise of control over the environment of a system.
- AJAX: A platform (composed of other platforms) that enables interactive web applications.
- CORBA: An object platform that enables the remote invocation and event architectural styles.
- Eclipse RCP: A Java language platform for software development tools and for applications generally.
- OSGi: An object platform originally designed for remotely managed software on small hardware
- J2EE Application Server: Oracle WebLogic Server, IBM WebSphere software platform, and many others
- Microsoft .NET
- RTOS: Nucleus PLUS, INTEGRITY, Tornado, and many others
- Athlon 64 X2, ARM 11, PowerPC G5, 8051, and many, many others

The specification of a platform may be a specification of the interface that platform provides for applications or may be a specification of the implementation of that platform.

Note also that platforms may be described as MDA models which then utilize other

platforms. For example, a CORBA implementation may use the JAVA platform which may then use an Intel I5 processor platform [43].

#### **4.8 MDA model transformation and execution**

A featured value proposition of MDA is automating the path from stakeholder-focused models to executable information systems. By automating the path from high-level models to executable information systems, we reduce the time, cost, and risk of producing and maintaining those systems while improving their fitness for purpose. Not all models are intended for or suitable for automation to execution – The models need to be sufficiently detailed and precise such that all the business requirements for the information systems are expressed in the models. MDA serves to separate the concerns between those business system requirements and the technology that implements them. This separation of concerns is achieved by using parameterized patterns to define how a business-focused model is transformed into a technology solution [43].

##### **Automating the path from models to executable systems**

There are two primary approaches to automating the path from models to executable systems. These are:

- A transformation pattern is applied to the model, producing technology specific artifacts or models. For example, a business information model may be transformed to XML Schema and/or C# or Java code.

- A model execution engine, implemented in some platform technology, directly executes the model – treating it as interpretable source code. That engine may then be able to produce or consume XML messages.

Both the transformation pattern and the execution engine achieve the same result – what was specified in model form becomes usable as an executable system, or perhaps a system simulation [43].

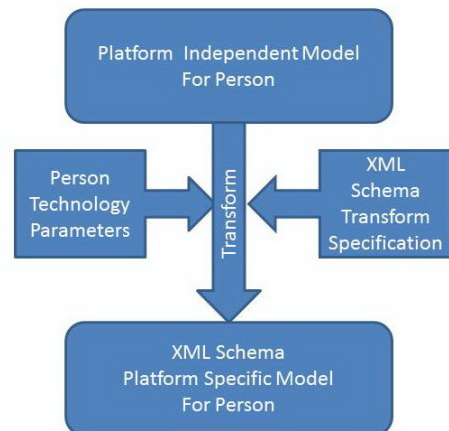
### **Platform specific and independent models**

Previously we defined the concept of a “Platform”. Some platform is required for any information system to execute. The implementation derived from transformation or an execution engine is specific to such a platform. When a model of a system is defined in terms of a specific platform it is called a “Platform Specific Model” (PSM) [52]. A model that is independent of such a platform is called a “Platform Independent Model” (PIM) [52]. MDA automates the production of a PSM model (that is closer to the technology) from a PIM, which is closer to business concepts and requirements. A transformation specification (which may also be a model) specifies how a PIM is transformed into a PSM based on parameters provided by developers. The parameters specify what models are transformed to what technology using specific technology choices [52, 43].

### **Example Transformation Pattern**

For example, an information model about a “Person” is platform independent – that is, it could be realized by any number of platform technologies. The model transformation is parameterized, selecting XML Schema as the platform and a specific XML “style” and naming rules. The XML schema pattern would have parameters defined for these XML specific choices (e.g. when to use elements vs. attributes) [43].

The diagram on the right illustrates this transformation pattern.



- “Platform Independent Model for Person” – the logical information model representing person information.
- “XML Schema Transformation” – a reusable component that executes the transformation, this component is specific to XML schema but allows for some options.
- “Person Technology Parameters” – a small specification that selects XML schema as the desired technology and supplies any parameters required by the transform component.
- “Transform” the actual transformation which produces the XML schema PSM for person.
- “XML Schema Platform Specific Model for Person” – a XML schema representation for Person as defined by the PIM [43].

Note that the same PIM may be parameterized differently and thus support different technologies. Multiple technologies could be used in the same system (e.g. for XML schema, SQL, and a user interface) or across different systems (e.g. supporting Java and C#). It should be noted that the produced executable solution may or may not be complete.

In some cases, MDA is used for just a portion of the system (e.g. the definition of business processes or component interfaces) and other implementation components are added using traditional languages. Other MDA solutions generate entire systems or system components [43].

### **Considerations for automating the path from models to executable systems**

The following are some of the advantages and overheads associated with automating the path from models to executable systems:

- Business stakeholders are more able to engage in and validate a business focused model, thus improving the resulting system's fitness for purpose. This does require effort on the part of business stakeholders to understand the business focused models.
- Once the transformation specification for a platform is created, it can be reused many times and for a variety of PIMs.
- Optimization can be enacted once in a transformation, to benefit all systems produced by that transformation.
- Transforms can be subject to careful scrutiny for cybersecurity and reliability; these features will then be passed on to the PSM.
- The reuse of the transformation leverages developer output, reducing coding time and cost. There may be an initial investment in producing the transforms for a specific platform, if one does not already exist for the platforms of interest.



- Due to the use of patterns, the produced system is very consistent and maps directly to the PIM requirements. If the transform is correct, the system will execute the PIM correctly.
- Investment in PIMs can be leveraged as follows:
  - Multiple technologies can be supported from the same PIM: across system components, across systems, and across time.
  - Changes in business requirements, as represented in the PIM, can be easily propagated to all PSM components.
  - Changes in technologies can be reflected in the transforms and then easily propagated to all PSM components [43].

#### **4.9 System lifecycle support in MDA**

MDA has an impact on a System's "System Development Life Cycle" (SDLC). A SDLC is the process that starts with a system initiative and results in a solution. A properly defined SDLC is important for managing any systems effort, but is crucial for large enterprise and government systems. The following summarize some of the impacts MDA can have on the SDLC [43]:

- Stakeholders are more engaged in understanding and validating the system. More effort is placed on producing an accurate and complete model. Optimally, MDA starts with requirements gathering and understanding the business needs.

- More effort is placed “up front” in defining or tuning and testing the transforms for a specific set of platform technologies. However early iterations may use less robust or tuned transforms.
- These up-front investments yield returns as system requirements evolve over time, thus supporting an agile methodology
- Components defined in models have clear boundaries and interfaces that developers can “code into” and quality control can validate
- Project resources can be more easily divided between a “PIM Team” and a “PSM Team” that can work somewhat independently, yet when combined produce a robust solution.
- Much less time is spent on coding and debugging.
- A system may be simulated prior to committing to full implementation. Simulation helps evaluate side effects, performance and user satisfaction.
- Changes are more easily managed, and can even be encouraged.
- Model based designs can also help automate the quality control and software assurance processes.
- The more structured MDA-SDLC reduces risk and unnecessary divergence internal to a system.

The net result is that a system’s effort moves and requires somewhat different expertise.

Once a team has mastered the MDA process they can function much more effectively as a “software development machine” with very reliable and repeatable results. Use of MDA is not an all-or-nothing proposition – initial MDA efforts may focus on the systems structure

and interfaces while still using more manual techniques within components. More mature MDA teams can then automate more of the system as experience and confidence grows [43].

## **CHAPTER V: METAMODELLING**

### **5.1 Problems in information system development**

The demands for more sophisticated systems have been relentless. Of course, the more sophisticated the requirements of a system are, the larger and more complex the deployed system is likely to be. Increased system complexity typically brings with it the following problems [19]:

- longer development times;
- more complex assembly due to number of components and number of people involved;
- increased cost and time for testing;
- increased maintenance costs.

Overall, this results in an increased time to market for any system, and increased development and maintenance costs in order for there to be any confidence that the quality of the system is not compromised. For software systems, as well as the problems outlined above which relate to the fundamental increase in lines of code, there is an additional qualitative difference to the systems being developed today compared to those of decades past. Modern systems are increasingly distributed in nature, as demonstrated by the ubiquity of enterprise applications. This adds another dimension to software complexity, and brings added challenges of communication and security to those listed above [19].

Life would be much easier if there was only one programming language and one deployment platform, but of course this is not the case, and for very good reasons. Diversity is not really a single challenge, but a category of challenges [19].

### **Diverse Domains**

The requirements of large systems often relate to a variety of domains that need to be reconciled by different stakeholders.

### **Diverse Customer Requirements**

The 'one size fits all' approach to software and systems is increasingly inappropriate in today's market. Vendors who offer products that can be tailored to the specific needs of a customer have a strong commercial advantage, but developing products that are truly customizable such that they can meet the demands of a broad customer base is a far from trivial matter.

### **Diverse Implementation Technologies**

Systems are often deployed across a number of different implementation technologies which need to be integrated, or need to be deployed for a number of separate implementation technologies.

Nowadays, "change is the only constant". Nearly all systems evolve over time. Typical reasons for this are [19]:

- change in customers requirements or market trends;
- change in implementation technologies or deployment platforms;
- support for additional functionality and features;
- availability of more effective implementation solutions;

- bug fixes.

## **5.2 Language driven development**

One of the distinguishing features of being human is our use of language. Languages are fundamental to the way we communicate with others and understand the meaning of the world around us. Languages are also an essential part of systems development. Developers use a surprisingly varied collection of languages. This includes high-level modelling languages that abstract away from implementation specific details, to languages that are based on specific implementation technologies. Many of these are general-purpose languages, which provide abstractions that are applicable across a wide variety of domains. In other situations, they will be domain specific languages that provide a highly specialised set of domain concepts. In addition to using languages to design and implement systems, languages typically support many different capabilities that are an essential part of the development process. These include: *Execution, Analysis, Testing, Visualization, Parsing, Translation, and Integration* [19].

### **5.2.1 Abstraction**

Abstraction has long been used as a means to allow humans to cope with complexity. Abstraction concerns distilling the essential characteristics of something relative to a particular perspective of the viewer. The two key ideas here are that some non-essential details are ignored, and that a particular context needs to be defined in order for the abstraction to make sense. Often abstraction involves the ‘chunking’ and organization of

information in a particular problem domain in order to allow the viewer to better comprehend the problem, by separating concerns. It is this information chunking that is the fundamental means for overcoming the limited human capacity for complexity, and languages are the means for capturing abstractions [19].

We highlighted in section 5.1 that diversity lies at the heart of modern system development. Going a step further can be stated that the challenge really boils down to a diversity of languages [19]:

- specialists require languages to address the particular facets of the problem that lie within their domain - often within each specialist domain there are numerous languages, with new languages being developed all the time;
- there are countless implementation languages - some differences are due to the continuing trend of increasing abstraction, some are due to the fact that different paradigms or individual languages are better suited to a particular problem-solution pair than others, and some are simply down to competitive commercial interests;
- the languages and syntax that capture the artefacts created during the development lifecycle.

Rather than trying to subdue this diversity by forcing everyone to talk (or model) using the same language, we should embrace it and allow everyone to use whatever language best suits their needs. In many cases, this may be a general purpose modelling or programming language, as these will be widely supported by tools and techniques, but in some cases more specialised languages may be more appropriate. An example of this might be an inventory-based system, where developers consistently have to express their models in

terms of inventory type concepts such as resources, services and products. By allowing engineers and domain experts to express themselves in the languages that they are both most comfortable with and that will give them the most expressive power, productivity can increase with corresponding gains for industry [19].

The argument against this is that by having a single standard language, there is only one language for developers to learn, so everyone can communicate more easily, and interoperability between tools will be much simpler. Whilst this is undoubtedly true, in order to make a modelling language that suits the needs of everybody (not just software engineers), it will suffer from the following problems:

- it will necessarily be a very large, bloated language;
- there are often contradictory needs of a language from different domains that cannot be reconciled in a single language;
- any gain made in widening the applicability of a language to different domains will be at the expense of the richness of the language that makes it so suitable for a particular domain.

### **5.2.2 Integration**

Language integration between two languages means that some or all of the language constructs of each language are in some way mapped to corresponding constructs of the other language. Some common applications of language integration are highlighted below [19]:

#### **Transformation**



The most publicised application of language integration is that of transformation, where an artefact in one language is transformed into an artefact of another. This type of activity is of prime importance in the Model Driven Architecture approach to information system development.

### **Artefact Integration**

If a system is comprised of a number of subsystems from different domains, and these different system aspects are described in different languages, then by integrating the languages, those aspects can themselves be weaved together to form a unified view of the system. This is typically used to integrate language artefacts that are at a similar level of abstraction.

### **Equivalence Verification**

Sometimes it is important to check whether an artefact written in one language is equivalent to one written in another. For example, it may be important to check whether an implemented system conforms to a precise system specification written in a high-level language. Again if the two languages are integrated appropriately, then this will be possible.

### **Synchronisation**

Language integration can also enable language artefacts to be synchronised. For example, whilst in some cases it might be appropriate to generate code for a system from a high-level model as a one-shot activity, in many cases it is desirable to keep the model and code in step. Similarly, if you have a diagramming language that allows graphical representation of a modelling languages, it is important to keep the graphical entities in step with any changes to the model.

### **5.2.3 The complete solution**

We described how languages can provide the overall solution to the challenges described in section 5.1 - more specifically an integrated framework of semantically rich, flexible and evolvable languages appropriate to their needs. This Language- Driven Development framework will:

- allow the construction of agile abstractions that are resistant to change;
- enable those abstractions to be transformed into, integrated with, validated against or synchronised with abstractions written in other languages;
- support powerful applications (editors, analysis and simulation tools, the aforementioned transformers and integrators) to be written and applied on those abstractions [19].

The right languages enable developers to be significantly more productive than using traditional development technologies because engineers and domain experts can speak in the languages they understand. Rather than dealing with low level coding issues, developers can use powerful language abstractions and development environments that support their development processes. They can create models that are rich enough to permit analysis and simulation of system properties before completely generating the code for the system, and are more reusable and agile. They can manipulate their models and programs in significantly more sophisticated ways than they can code. Moreover, provided the language definitions are flexible, they can adapt their languages to meet their development needs with relative ease. Language-Driven Development is the next generation development paradigm which can provide a step gain in productivity through the recognition that

languages, rather than objects or models, are the abstractions needed for today's development environment [19].

### **5.3 Features of Languages**

Whilst the nature, scope and application of the languages used in systems development is naturally diverse, there are a number of key features they all share. Understanding these features is a first step towards developing a generic approach to modelling languages [19].

#### **Concrete Syntax**

All languages provide a notation that facilitates the presentation and construction of models or programs in the language. This notation is known as its concrete syntax. There are two main types of concrete syntax typically used by languages: textual syntax and visual syntax. A textual syntax enables models or programs to be described in a structured textual form. A textual syntax can take many forms, but typically consists of a mixture of declarations, which declare specific objects and variables to be available, and expressions, which state properties relating to the declared objects and variables. The following Java code illustrates a textual syntax that includes a class with a local attribute declaration and a method with a return expression [19]:

```
public abstract class Thing
{
    private String nameOfThing;

    public String getName()
    {return nameOfThing;}
}
```

}

A visual syntax presents a model or program in a diagrammatical form. A visual syntax consists of a number of graphical icons that represent views on an underlying model. A good example of a visual syntax is a class diagram, which provides graphical icons for class models [19].

### **Abstract Syntax**

The abstract syntax of a language describes the vocabulary of concepts provided by the language and how they may be combined to create models. It consists of a definition of the concepts, the relationships that exist between concepts and well-formedness rules that state how the concepts may be legally combined [19].

### **Semantics**

An abstract syntax conveys little information about what the concepts in a language actually mean. Therefore, additional information is needed in order to capture the semantics of a language. Defining a semantics for a language is important in order to be clear about what the language represents and means. Otherwise, assumptions may be made about the language that lead to its incorrect use. For instance, although we may have an intuitive understanding of what is meant by a state machine, it is likely that the detailed semantics of the language will be open to misinterpretation if they are not defined precisely. What exactly is a state? What does it mean for transition to occur? What happens if two transitions leave the same state. Which will be chosen? All these questions should be captured by the semantics of the language [19].

### **Mappings**

In the real world, languages do not exist in isolation. They will have a relationships to other languages. This may be via translation (concepts in one language are translated into concepts in another language); semantic equivalence (a language may have concepts whose meaning overlaps with concepts in another language) or abstraction (a language may be related to another language that is at a different level of abstraction). Capturing these relationships is an important part of a language's definition as it serves to place the language in the context of the world around it. Furthermore, mappings exist between the internal components of languages, such as between a concrete and abstract syntax, and are an important part of a language's architecture [19].

#### **5.4 What is a Metamodel?**

In its broadest sense, a metamodel is a model of a modelling language. The term "meta" means transcending or above, emphasising the fact that a metamodel describes a modelling language at a higher level of abstraction than the modelling language itself. In order to understand what a metamodel is, it is useful to understand the difference between a metamodel and a model. Whilst a metamodel is also a model, a metamodel has two main distinguishing characteristics. Firstly, it must capture the essential features and properties of the language that is being modelled. Thus, a metamodel should be capable of describing a language's concrete syntax, abstract syntax and semantics. Secondly, a metamodel must be part of a metamodel architecture. Just as we can use metamodels to describe the valid models or programs permitted by a language, a metamodel architecture enables a metamodel to be viewed as a model, which itself is described by another metamodel. This

allows all metamodels to be described by a single metamodel. This single metamodel, sometimes known as a meta-metamodel, is the key to metamodelling as it enables all modelling languages to be described in a unified way [19].

## **5.5 Why Metamodel?**

As discussed previously, system development is fundamentally based on the use of languages to capture and relate different aspects of the problem domain. The benefit of metamodelling is its ability to describe these languages in a unified way. This means that the languages can be uniformly managed and manipulated thus tackling the problem of language diversity. For instance, mappings can be constructed between any number of languages provided that they are described in the same metamodeling language.

Another benefit is the ability to define semantically rich languages that abstract from implementation specific technologies and focus on the problem domain at hand. Using metamodels, many different abstractions can be defined and combined to create new languages that are specifically tailored for a particular application domain. Productivity is greatly improved as a result.

Metamodels have been around for many years in a wide variety of different application domains and under various pseudonyms: "data model", "language schema", "data schema" are all terms we have seen. Wherever there is a need to define a language, it is common to find a metamodel. This is particularly the case for standards, which by virtue of being a standard must have a precise definition. Examples include AP233 and SysML (systems engineering), SPEM (process modelling), OSS (telecoms) and CWM (data warehousing).

The Object Management Group (OMG) has been particularly involved in their use in the standards arena. One of the largest metamodels (about 200 pages long) is contained in the UML specification. With the advent of MDA and the increasing need for standardisation across the systems development community, the number of applications of metamodels is set to grow significantly [19].

## **5.6 Metamodelling languages**

A metamodel is written in a metamodelling language, which is described by a metamodel. As described above, the aim is that the same metamodelling language (and meta-metamodel) is used to describe any number of different languages. Thus, provided that the modelling languages have been defined in the same metamodelling language, it is possible to treat their definitions in a unified manner. For example, they can be stored and manipulated in the same way, or related by mappings. What distinguishes a metamodelling language from a general purpose programming language like Java or a modelling language like UML? The answer is that a metamodelling language is a language specifically designed to support the design of languages. An essential requirements of a metamodelling language therefore is its ability to concisely capture all aspects of a modelling language, including its syntax and semantics [19].

The task of creating a metamodel for a language is not a trivial one. It will closely match the complexity of the language being defined, so for example, a language containing rich executable capabilities will be much more complex to define than a simple static language. However, there is a clearly defined process to constructing metamodels, which does at least

make the task a well-defined, if iterative, process. The process has the following basic steps

[19]:

- defining abstract syntax
- defining well-formedness rules and meta-operations
- defining concrete syntax
- defining semantics
- constructing mappings to other languages



## **Part II: PRACTICAL STUDY AND IMPLEMENTATION**

In the second part of this work we present, analyze and implement the process automation principles described in part one using two different frameworks, Bizagi BPMN suite and MOSKitt. The process that will be considered to be automated is the User Access Management process and this process is modelled and automated in both frameworks; Bizagi BPMS suite and MOSKitt. In order to fully understand the process and its operations specific and detailed face to face interviews are performed with each of the stakeholders involved in the process. The interviews are a vital part of the study since they help to properly capture and model the business process. Practical applications on how to automate the process in each of the frameworks are performed. This try, learn, error and improve process enabled the complete understanding of both frameworks taken into consideration and facilitated the process automation of the User Access Management process in both frameworks. The details on how to automate the process in each framework are presented and finally a comparative study on the pros and cons of each solution is performed in order to evidence which one is more applicable in each case. This is a fundamental part of the practical applications of the study since enables us to suggest that Bizagi BPMS suite is a framework suitable for process automation and optimization for the telecommunications industry in Albania. We continue the second part by providing a complete overview of the Business Process Management practices in the telecommunication industry in Albania. All seven major operators in the telecommunication industry in Albania are considered and questionnaires and interviews are used in order to evidence the business process management practices in each of the

companies. The business market conditions of the telecommunication industry in Albania are presented along with the results on business process management practices. The results of the study show that Bizagi BPMS suite would be a very good match for business process management in the respective companies.

## **CHAPTER VI: RESEARCH METHOD AND METHODOLOGY**

### **6.1 Information sources**

The business process that we are taking into consideration for process automation is the User Access Management process present in Vodafone Albania, the leading telecommunications company in Albania. In order to familiarize with the process in consideration face to face interviews have been performed with the process stakeholders (Corporate Security and IT Support teams) involved. The business owner of the process is the Corporate Security section within Vodafone Albania which is responsible for the management of the process.

In order for an employee to gain access into a system the employee needs to perform a request specifying in it the details of the access needed, the system when access is needed, the period for which the access is needed, priority of the request and other useful information. The request goes through an approval cycle which includes the line manager of the employee and the Corporate Security manager. Corporate Security manager is an entity within the company which is responsible for the logical and physical security and all accesses into systems go through his/her approval. If the request is rejected from the line manager and/or Corporate Security manager the process ends and the employee is informed. Only when the request is approved by both the line manager and Corporate Security manager it goes for implementation to the respective administrator. In Figure 1 is presented the User Access Management process in the BPMN notation.

For the practical implementation of automating the above process using the model driven approach two frameworks have been taken into consideration, Bizagi BPMN suite and MOSKitt. Bizagi BPM Suite is one of the best alternatives available for fast, flexible process management. It is a Business Process Collaboration Platform that speaks the language of business process. From design and modeling to automation and deployment, Bizagi BPM Suite supports the complete BPM lifecycle to make continuous improvement a reality. The solution enables business experts design, document, execute and evolve their process model with complete confidence. Intuitive drag and drop, code-free updates and automatic document generation tools make this a seamless experience, even without technical knowledge [1]. In the MOSKitt framework we will focus our attention in “MOSKitt Code Generation” module which supports the semi-automatic generation of OpenXava [46] applications through a chain of model transformations from models (UML models, User Interface Models) to code. The model transformation functionalities provide automatic generation of the OpenXava code necessary to ensure the persistence of the entities in the database and the generation of the AJAX user interface. An enterprise web application with create, read, update, delete and export functionalities is automatically generated and operational. The User Access Management process is automated in both frameworks using the documentation available of the respective solutions.

In order to present the situation in the Business Process Management practices for the telecommunications industry in Albania questionnaires and face to face interviews have been performed with respective employees responsible for Business Process Management in all the seven major telecommunication companies operating in Albania. Also annual

reports provided from the National Regulatory Authority have been used in order to present the telecommunication market dynamics in the Albanian market.

## **6.2 Methods, basic instruments and population of the study**

Practical implementations are used in order to automate the User Access Management process in Bizagi BPM suite and MOSKiit. A constant practice with both solutions from installation to configuration to testing has concluded with the implementation of the process automation in both cases. In the first part it has been performed a pure implementation approach. The instruments used in this scenario are the tools taken into consideration for implementation; Bizagi BPM suite and MOSKit framework. A comparative approach taken for each of the solutions is performed highlighting which is best for each specific scenario.

The telecommunications sector in Albania is dominated from seven major operators that have an across the country presence. These operators are Vodafone Albania sh.a., Albanian Mobile Communications sh.a, Albtelecom sh.a., Plus Communication sh.a., Abcom sh.p.k., Abissnet and Albanian Satellite Communications (ASC). As per the latest report published from the National Authority of Electronic and Postal Communications there are around 100 operators present and operating in Albania. Only the above seven mentioned operators have an across the country presence while the other operators are very small and have only local or regional presence [40]. Interviews and questionnaires have been used with the respective business process management employees working in the above companies. In this case the results presented are 100% accurate since they are not based on

a sample of the population but from the analysis results of the entire population in consideration in the study.

### **6.3 Data management and elaboration**

In order to collect information regarding the practices present in the telecommunication industry in Albania an online questionnaire has been used using google forms and has been sent for completion to respective employees in the companies into consideration responsible for business process management. In some cases interviews have been performed as well in order to better clarify some points. Below is the questionnaire used for the survey, using screenshots from the google forms. The questionnaire is composed of the below main areas:

- Business Process Modelling and Business Process Management awareness in the companies
- Responsible team within the company for Business Process Management
- Tools and formalisms used for business process modelling, particular focus on the use of BPMN
- Tools used for business process management, particular focus on Bizagi BPM suite
- Tools used to simulate business processes



## Survey on Business Process Management in Albania

Thank you for your effort and support on completing the below quick survey. The aim is to gather information regarding the applicability of Business Process Management principles within the companies operating in Albania, evidencing good practices and providing a general overview on the state of Business Process Management in Albania. Confidentiality of the responses will be assured at all stages of the process.

**Please complete the name of the company.**

Company Name

**Please provide your name and position within the company.**

Name Surname; Position

**Are you aware of Business Process Modelling?**

**Are you aware of Business Process Management?**

**Does your company has a dedicated team for Business Process Management/Business Process Improvement?**

**If yes, please provide a brief description of the responsibilities of the team?**

**If no, is someone within the company responsible for Business Process Management?**

Role responsible if any, otherwise no

**What solution/tools does your company (Business Process Management team) use to model business processes?**

Name of the solution/tool



**What formalism (language) does your company (Business Process Management team) use to model business processes?**

**Does your company (Business Process Management team) use BPMN language to model business processes?**

**What solution/tools does your company (Business Process Management team) use for Business Process Management?**

Name of the solution/tool

**What tools does the Business Process Management team use to automate Business Processes, Workflows?**

Does the above mentioned business process management tool provide comprehensive performance historical reports?

If yes, please provide some details on the type of reports provided.

Does the Business Process Management team use historical reports from the above tool in order to view trends and propose process improvements?

Does the Business Process Management team use tools to simulate business processes in order to improve and optimize them?

If yes, please provide the name of the tool used.

Does your company use Bizagi BMPS suite for business process management?

Submit

Never submit passwords through Google Forms.

100%: You made it.

Powered by  
 Google Forms

This content is neither created nor endorsed by Google.  
[Report Abuse](#) - [Terms of Service](#) - [Additional Terms](#)

#### 6.4 Difficulties faced during data collection

There have been encountered several difficulties in order to obtain the information needed for the completion of the study. For the first part related to the practical implementation of the automation of the User Access Management process in both platforms Bizagi BPM Suite and MOSKitt, the majority of the problems faced refer to the incomplete documentation in some cases. Some particular errors have been encountered in the platforms (particularly in MOSKitt) for which special support has been required. The issues have been overcome by getting in contact with platform owners and developers for troubleshooting advice in the problematic cases. Considerable amount of time has been spent in specialized online forums of both platforms in order to advance knowledge and solve the issues encountered in order to complete the automation of the business process into consideration.

Regarding the survey on business process management practices in the telecommunication industry in Albania there were initial problems on identifying the respective persons responsible for business process management in the companies in scope of the study. The contacts have been identified through extensive personal networking and using social media application. Once the contacts have been identified they have been presented with the questionnaire. In some cases the completion of the questionnaire has taken a long period of time and also has not been complete. In order to solve this issued interviews have been used in order to clarify the information needed for the questionnaires.

## CHAPTER VII: BIZAGI BPM SUITE

### 7.1 Overview

Business Process Management (BPM) is a corporate management philosophy. Its main objective is to provide company directors and process owners with the right information elements for the proper allocation of the organization's resources. This will increase their efficiency and profitability by means of the systematic management of business processes that must be continuously modeled, automated, integrated, monitored and optimized [1].

Bizagi BPM Suite is one of the best alternatives available for fast, flexible process management. By bridging the gap between IT and business, Bizagi empowers organizations to get the results they want. It is a Business Process Collaboration Platform that speaks the language of business process. From design and modeling to automation and deployment, Bizagi BPM Suite supports the complete BPM lifecycle to make continuous improvement a reality. The solution enables business experts design, document, execute and evolve their process model with complete confidence. Intuitive drag and drop, code-free updates and automatic document generation tools make this a seamless experience, even without technical knowledge [1].

There are three main components in the Bizagi BPM Suite as shown in Figure 12: *Bizagi Modeler*, *Bizagi Studio*, and *Bizagi Engine* that help business analysts manage the complete life-cycle of business processes [1].

- **Bizagi Modeler:** Bizagi Modeler is a freeware application, independent from Bizagi Studio that enables business analysts to visually diagram, model and document

business processes in industry-standard BPMN (Business Process Model and Notation).

- **Bizagi Studio:** Bizagi supports the model driven architecture approach to process automation and by following the “modeling over programming” philosophy, business experts have everything they need to transform process models into real, running applications and workflows. From defining the data model and UI to integrating IT assets and everything in between, Bizagi’s in-built wizard supports every step of the way.
- **Bizagi Engine:** Deployed on JEE or .NET to fit whatever architecture companies have in place, Bizagi Engine executes and controls the business processes automated by Bizagi Studio. “Update-once and change-everywhere” optimizes the workload up and down the value chain, reducing time and cost [8]. Bizagi’s unique multi-language portal makes pending activities easy to visualize, while KPIs provide total control over process performance



Figure 12: Bizagi BPM Suite components [23]

Bizagi Studio offers an integrated Process Modeler. However, it is recommended to diagram and document processes in Bizagi Modeler and then automate those processes with Bizagi Studio.

The first step to create solutions is to design the Process model or Process flow using Bizagi Modeler. The model, known as a chain of activities, is the fundamental structure of the project, to which variables and elements needed are included in accordance with organization's requirements. After designing the process model, the next step is to build the solution that is to automate the process model. To automate is to convert all the process activities into a technological application. Bizagi Studio is the construction environment used to automate the processes that were defined with the Bizagi Modeler. An easy-to-use wizard guides the business analyst through each of the automation steps: defining the data model, the user interface, the business rules, the work allocation and the integration with other applications, among other things. The automated model is stored in a database, and is interpreted and executed by Bizagi Engine. The model is presented in a Work Portal (web application) accessed through a browser. Bizagi Engine based on the process model previously built, watches for the correctness of the execution in the different tasks and activities that intervene in the business process; by controlling and verifying that the tasks are done in the correct moment, by the correct person or resource, and according to company's guidelines, objectives, and other fundamental rules. The resulting Work Portal has a very important characteristic: when the process is modified (any element of the model) it will automatically be updated and the change will be immediately displayed [1].

**Improve**

Bizagi has a complete set of performance reports and indicators about the processes which will allow business analysts to analyze business operation in real time and analyze historical performance information. These indicators enable process owners and business managers to gain insight into business operation and identify: bottlenecks, resource performance, service levels and trends. This information is the basis for process improvement [1].

## **7.2 Bizagi Modeler**

Bizagi Modeler is a business process modeling and documentation tool. The Modeler enables to visually diagram, model and document business processes in industry-standard BPMN (the modeler supports the current version, BPMN 2.0) and also is able to publish high quality documentation in Word, PDF, Sharepoint or Wiki. Processes can be easily imported from and exported to Visio or XML, and other tools. All processes are saved with a **.bpm** file extension. Each file is referred to as a model and may contain one or more diagrams. A model can refer to a whole organization, a department or a specific process depending on business needs. Multiple diagrams are positioned as individual sheets (tabs) within the model. It is possible to navigate between diagrams in the model by selecting the associated sheet tab located at the bottom of the model as in Figure 13 [9].



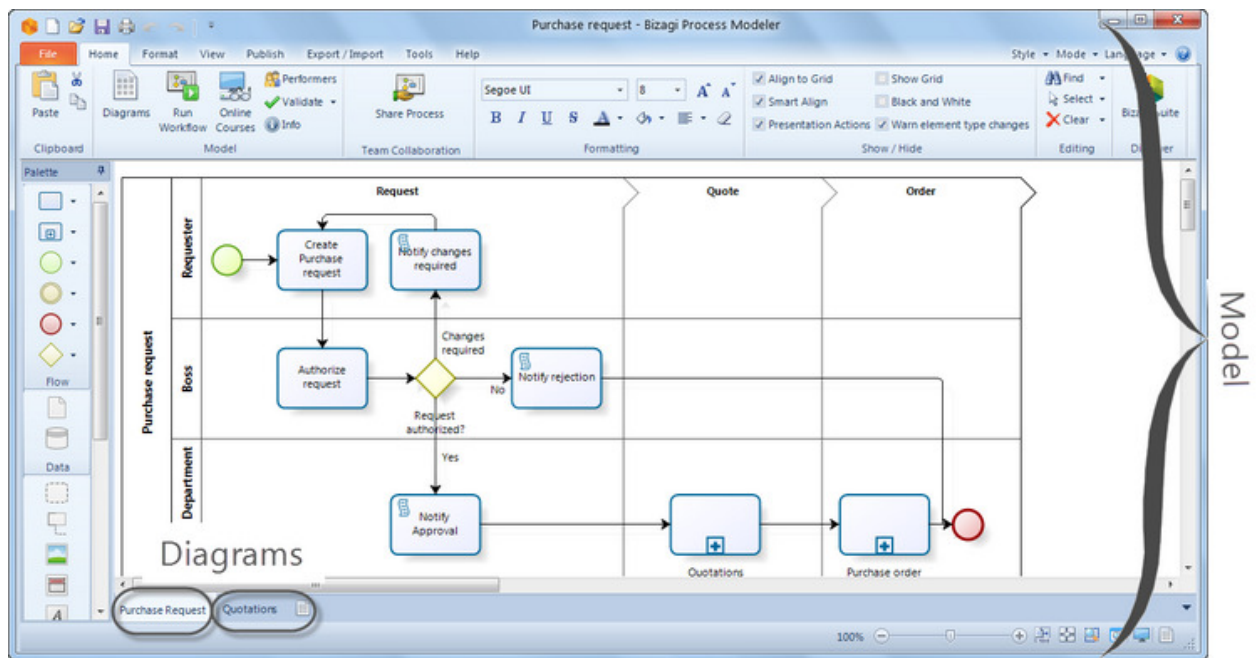


Figure 13: Bizagi process modeler

Bizagi Modeler is a freeware application that you can download from the internet and has a very simple, easy and intuitive interface. It has five main elements; *Toolbar*, *Ribbon*, *Palette*, *Element Properties* and *View* as explained in Figure 14 [9].

*Toolbar* - contains quick access commands to a subset of any menu within Bizagi Modeler.

*Ribbon* - contains the main controls to manage each Process Model. These are organized into different tabs.

*Palette* - contains the BPMN graphical elements used to define a process model.

*Element properties* - are used to document the process. Each element has its own properties and depending on the type of element, tabs are displayed with additional information and functionality.

View options - On the bottom right corner of Bizagi Modeler are view options that help to navigate through the process [9].

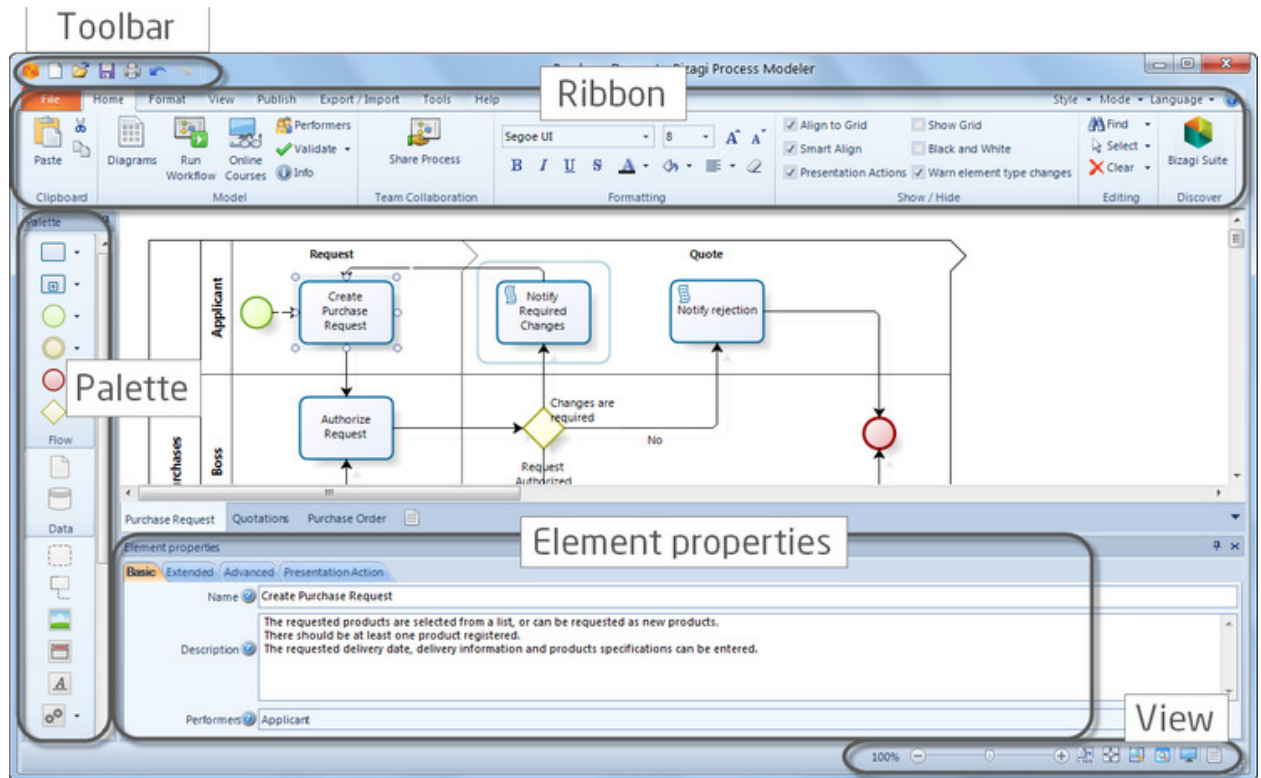


Figure 14: Bizagi user interface explained

### 7.2.1 Modeling a process in Bizagi Modeler

Modeling a process is an iterative and straightforward task to represent business conditions in a flow diagram, using symbols and shapes. To explain and go into detail how is possible to easily diagram a process; we will use a *Purchase Request Process*. The following are the steps to be carried out in the Process [9, 10]:

- A Purchase Request is created
- The employee's Immediate Supervisor (Boss) approves, rejects or changes the request

- Quotations are obtained in order to select a Supplier
- A Purchase Order is created
- The Administrative Manager approves, rejects or modifies the Order
- The Purchase Order is sent to the Supplier
- The Purchase Order is created in the ERP

As soon as you open the Modeler a Pool will be ready to start diagramming.

1. Name your pool. It is usually the name of the process you are about to diagram. To change the name of the Pool double-click on it, press *F2*, or right-click on it, and then select *Edit text* from the display menu as shown in Figure 15 [9, 10].

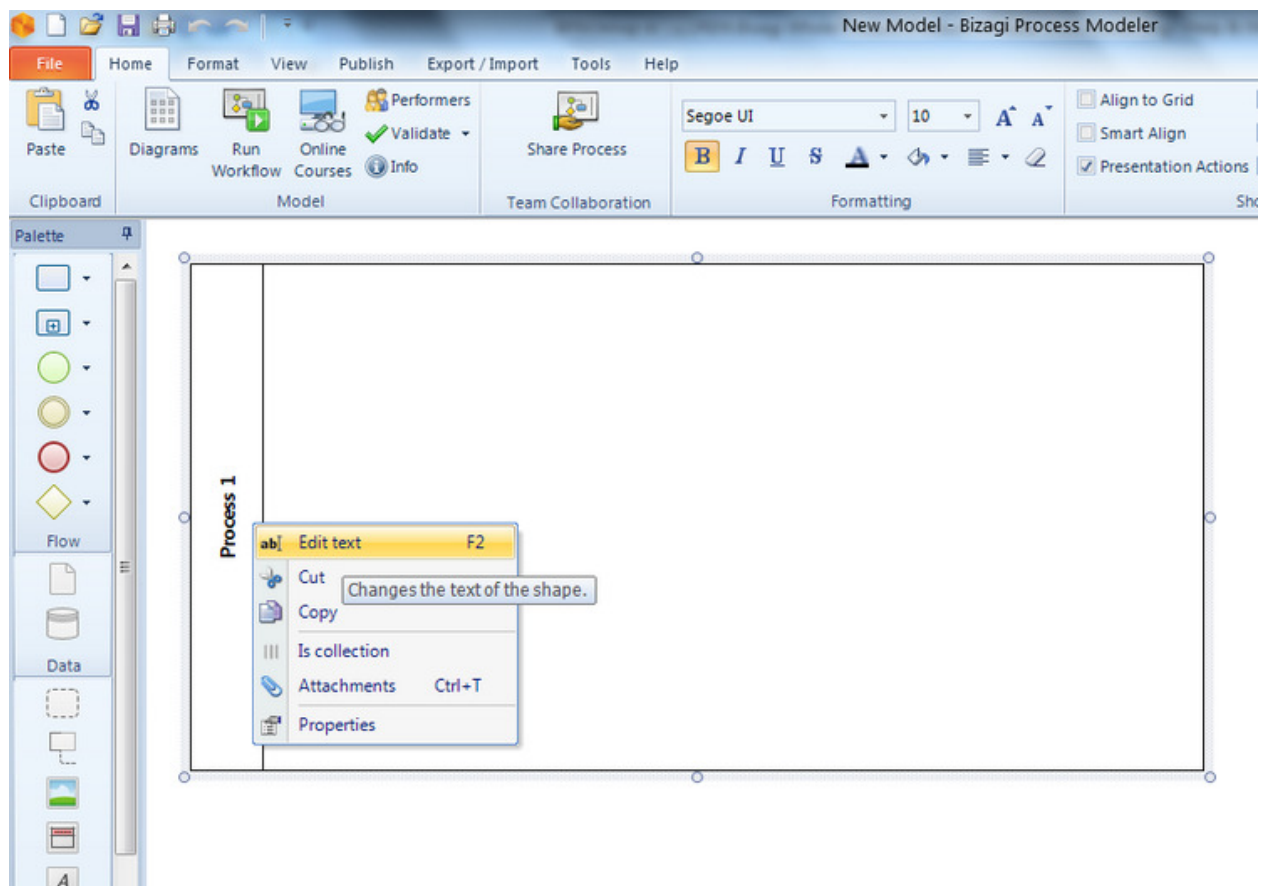


Figure 15: Provide the name of the Pool in Bizagi Modeler

2. Add a Lane to include participants in your process. Drag and drop a lane, for each participant, from the Palette as shown in Figure 16. In our example, we will select three lanes: One for the *Immediate Supervisor (Boss)*, one for the *Requester* and one for the *Purchase Department* [9, 10].

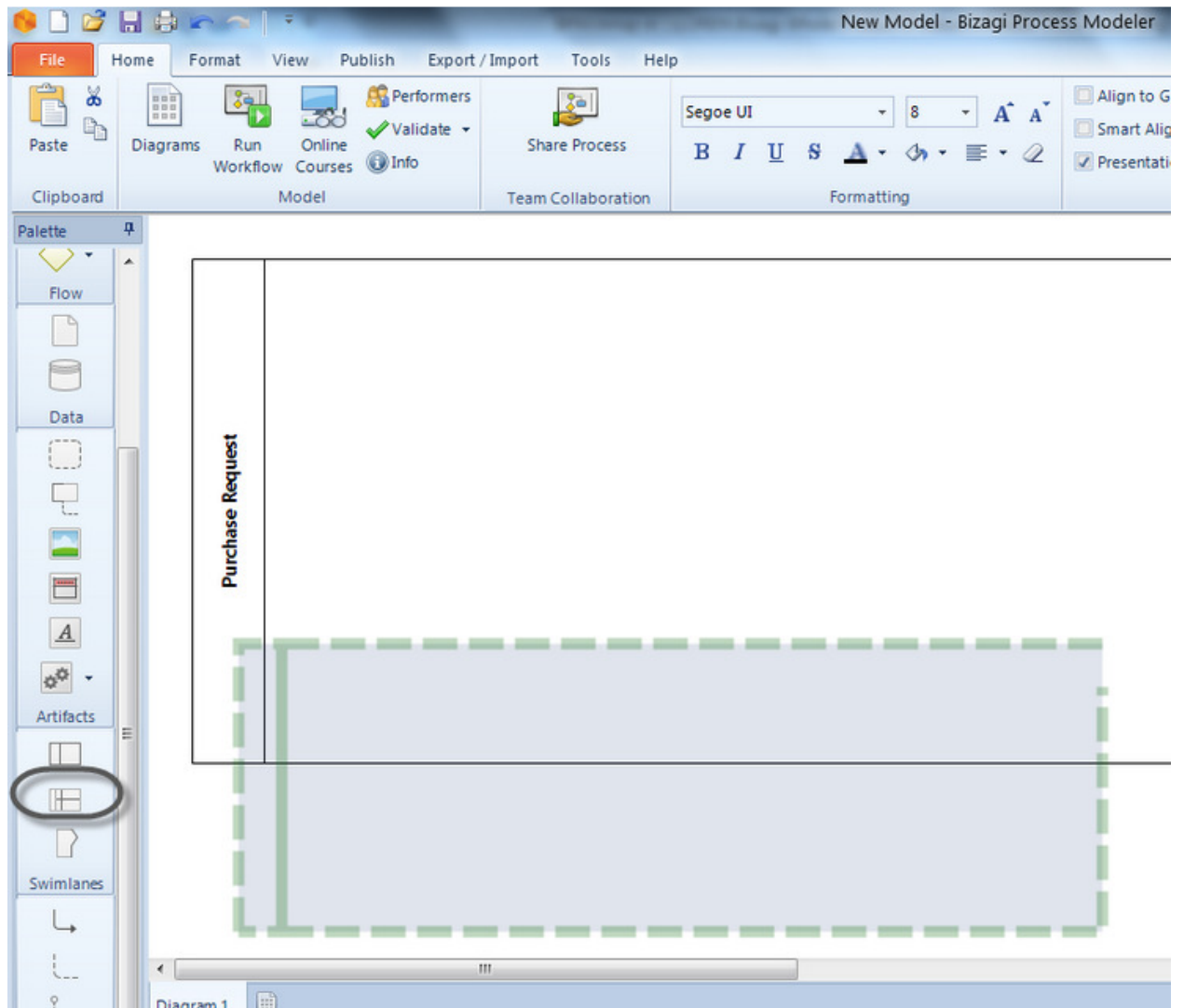


Figure 16: Drag and drop lanes from the Palette in Bizagi Modeler

3. Include a start point in your process. Drag and drop a Start event from the Palette as shown in figure 17.

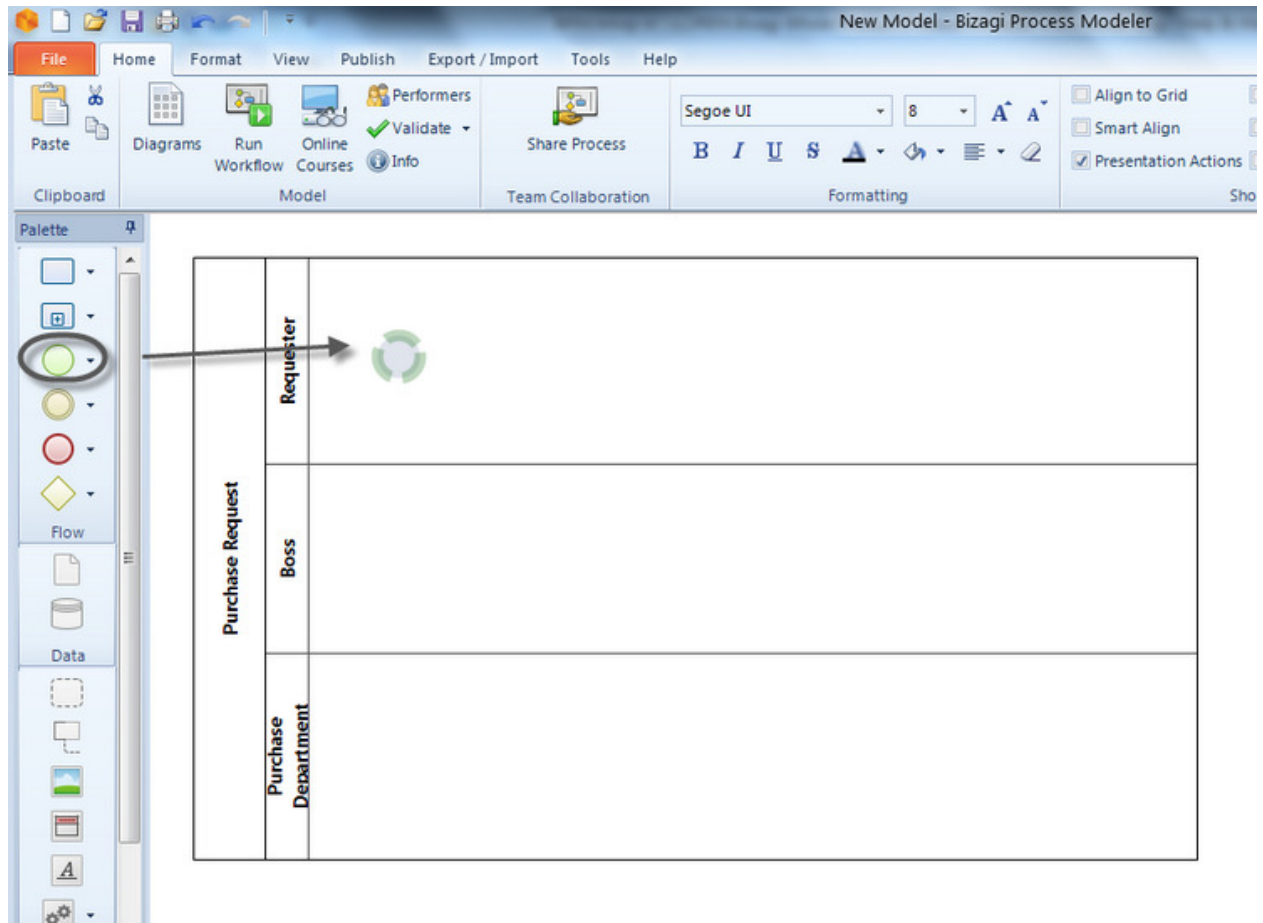


Figure 17: Drag and drop a Start event from the Palette in Bizagi Modeler

4. Continue diagramming the process using the Pie Menu. Select the next element and drag and drop where you wish to locate it as shown in Figure 18 [9].

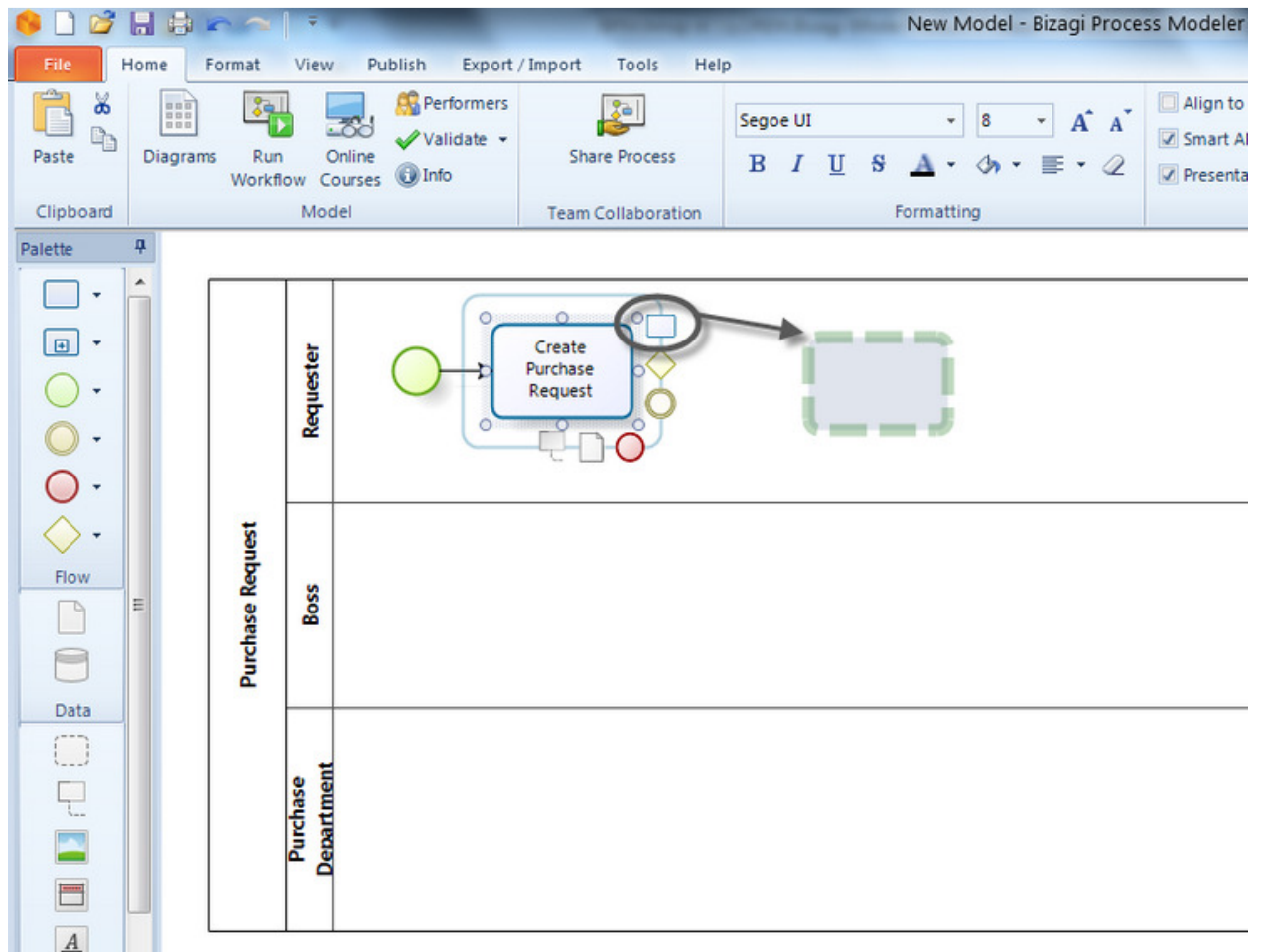


Figure 18: Diagramming the process using the Pie Menu in Bizagi Modeler

5. In order to connect two diagram elements in a sequence flow, select an object from the Pie Menu and drag it to the second diagram element. They will automatically connect as shown in Figure 19 [9].

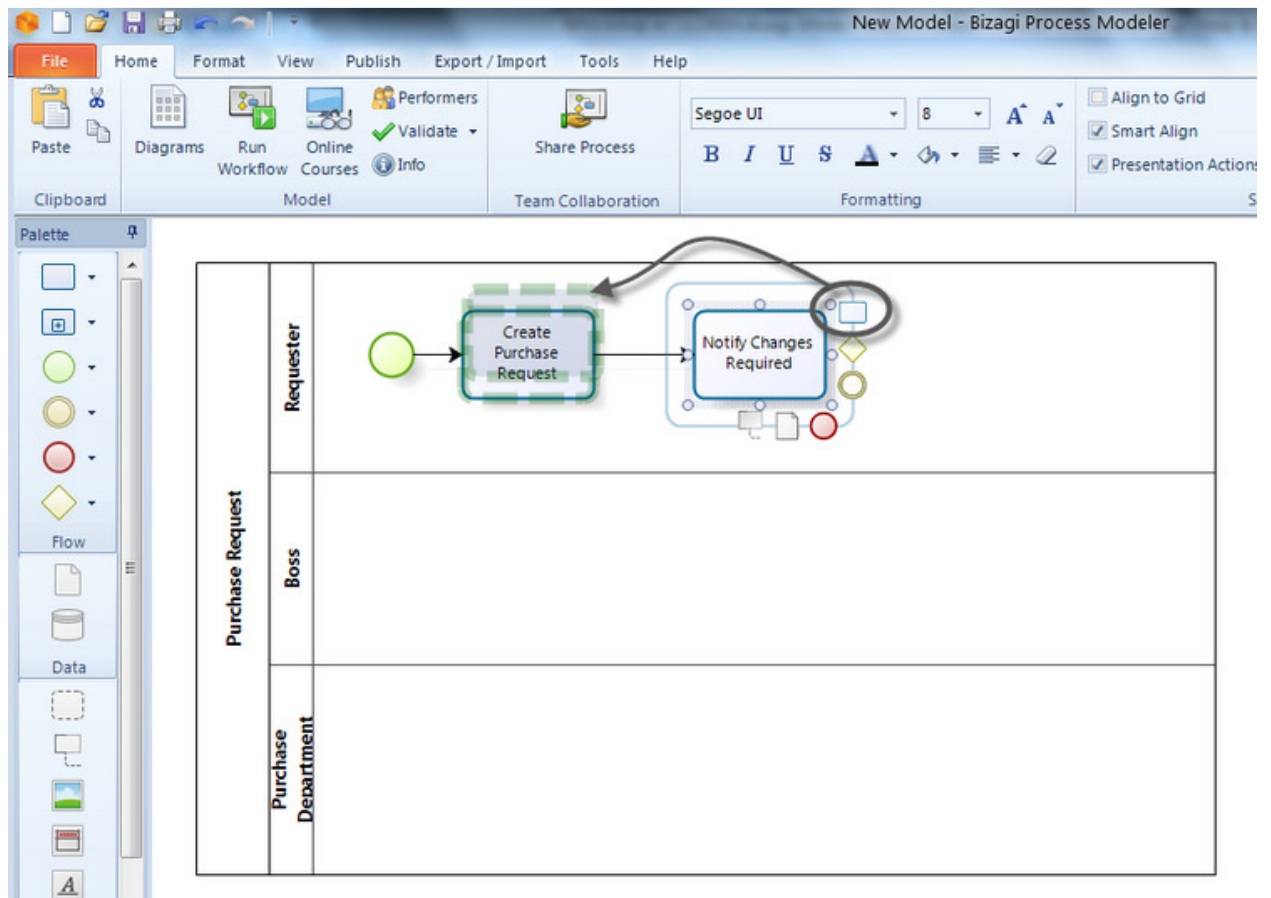


Figure 19: Connecting two diagram elements in a sequence flow in Bizagi Modeler

6. Continue selecting the required shapes until your diagram is complete.
7. To resize the pool, select and drag the appropriate corner of the border. In Figure 20 is displayed the basic diagram of the Purchase Request process [9].



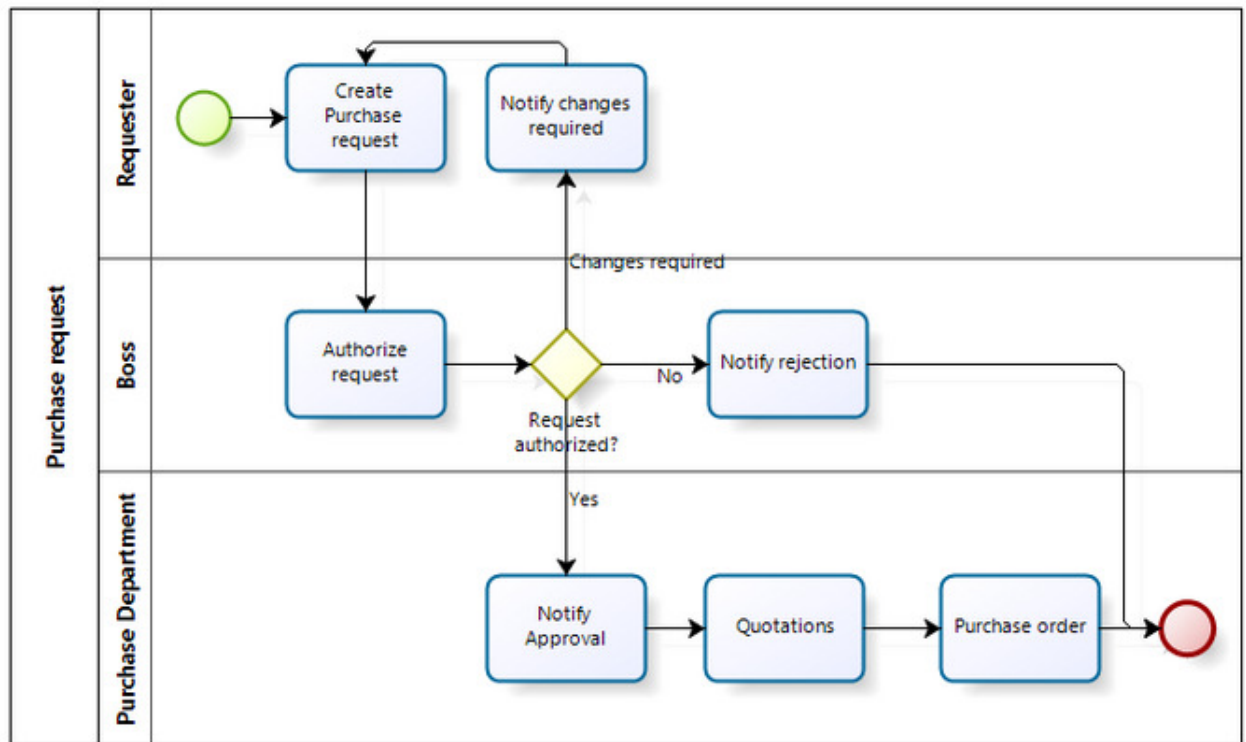


Figure 20: The basic diagram of the Purchase Request process

### 7.2.2 Creating a sub-process

A sub-process is a compound activity that is included within a process. Compound means that it can be broken down into lower levels, that each level includes shapes and elements within it. In the previous section we defined a task called “*Quotations*”, which is a sub-process (as we now realize that this task contains many activities) and we need to transform the diagram element and define the sub-process flow [9].

1. To transform the Task, (in this case *Quotations*), to a sub-process element, right-click on it and select Transform to sub-process from the display menu as shown in Figure [21].



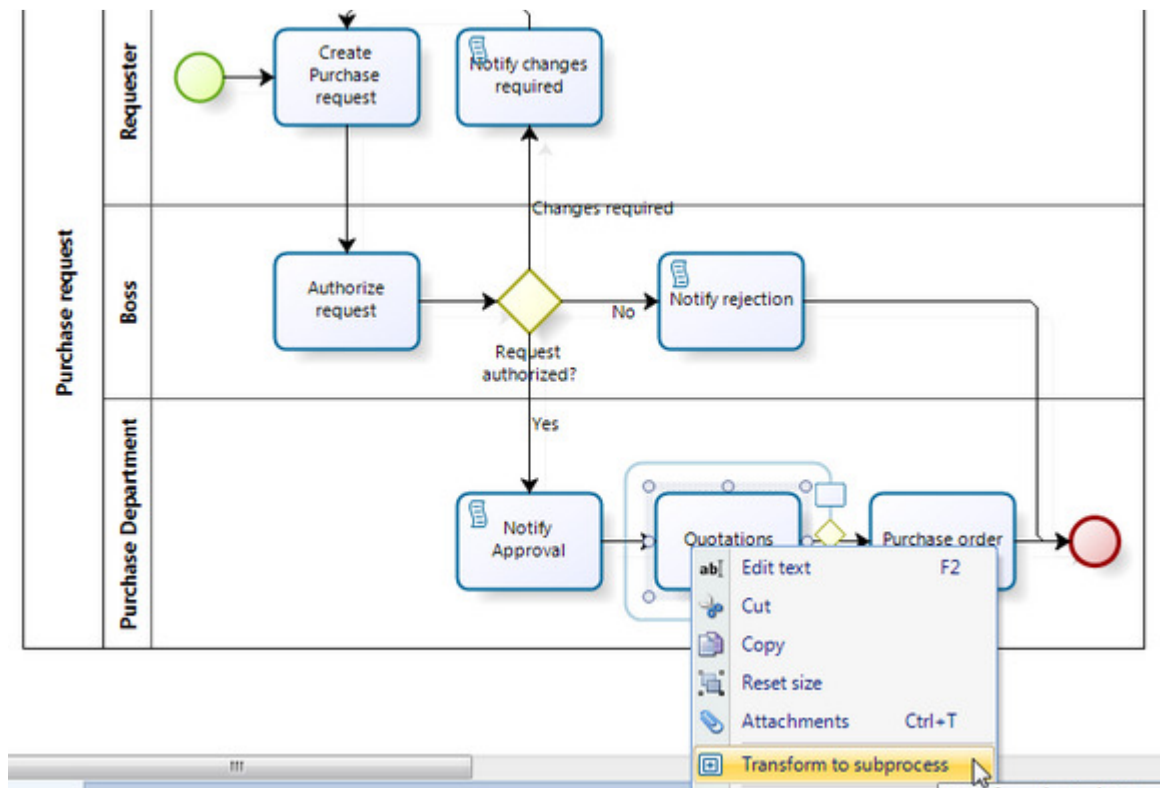


Figure 21: Transform the Quotations Task into a sub-process

2. Once the Task has been converted to a sub-process it is necessary to define its related diagram. Right-click on the sub process element and select the Edit sub-process from the display menu as shown in Figure 22 [9].

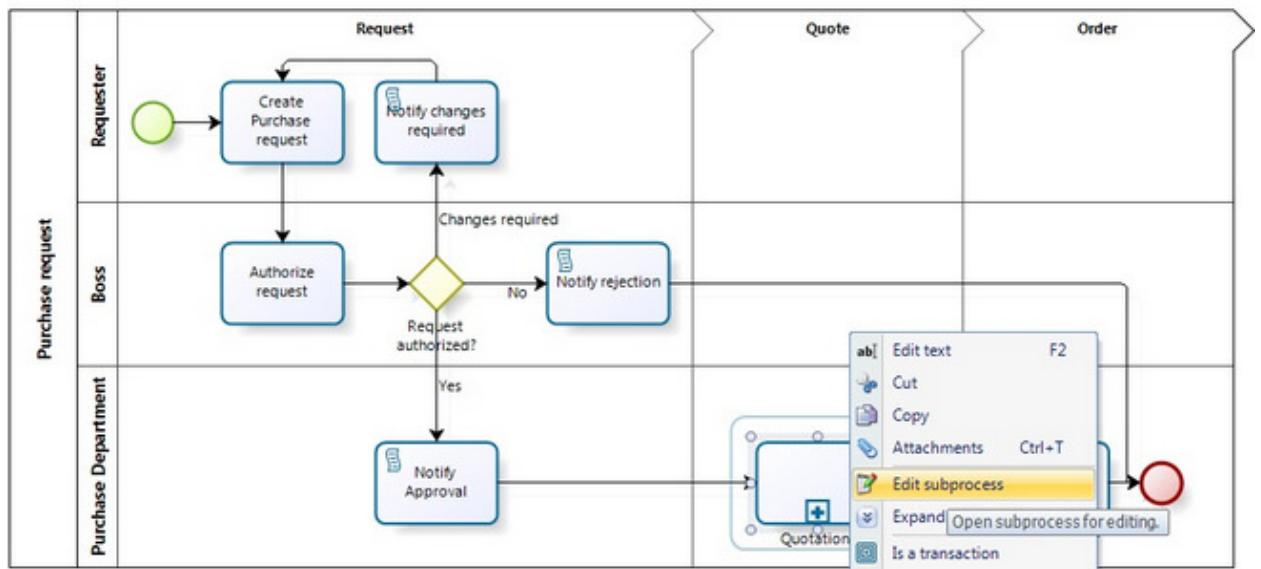


Figure 22: Define the diagram for a sub-process

3. This will automatically open a new diagram page to include the sub process information.

On this page we can diagram the associated sub-process, just the way we diagrammed the first process. Just drag and drop the elements as needed and place them where appropriate.

BPMN defines several types of sub-process that respond to particular business needs. By default sub-processes are created as embedded but you can change the sub-process type anytime [9].

### 7.2.3 Generating documentation, exporting and importing diagrams

Bizagi Modeler provides the possibility publish the process model complete documentation in any of the following formats and share with the organization [9, 11].

- Microsoft Word

- PDF
- Mediawiki
- Web file (opened through a browser)
- Microsoft Sharepoint

There is also possible to export process diagrams to other Modeling tools or exports customized attributes and reuse them in other Bizagi Process Models [9, 11].

- Microsoft Visio: export diagrams to Visio 2003, 2007 and 2010
- Image files: export diagrams to png, bpm, svg or jpg format
- XPDL: export your diagram to XPDL 2.1
- BPMN: export your diagram to BPMN 2.0 xml format
- Attributes: export your customized Extended Attributes and use them in other Bizagi Process Models, to maintain a standard in your documented processes.

Bizagi Modeler supports the importing of diagrams from Microsoft Office Visio or XPDL format files. Importing existing models will enhance agility and continuous process improvement efforts. Process diagramming requires a lot of dedication. Organizations often devote a significant amount of time and resources to it, and may use several diagramming tools. If this is the case and you have existing diagrammed Processes, all that valuable work will not be wasted. You can import your diagrams to the Bizagi Process Model and seamlessly continue to diagram and document [9, 11].

#### **7.2.4 Simulation**

Randomness is simulated by the use of probabilities for sequence flows and token routing and also by using statistical distributions to reflect variability in process times of activities

etc. To make sure results are valid, the simulation needs be run for long enough to yield random behavior without chance (consider the scenario of tossing a coin or rolling a dice multiple times). Provision should be made to compare results from the same scenario, but different run lengths or replications. The required run length to yield usable outcomes depends on the process model structure, amount of variability and the objective; consequently, a single recommended run length cannot be provided. A replication shares the same scenario configuration and runs for the same length of time, but uses an alternative random stream. Simulation is well known for providing what-if analysis capabilities; a single simulation run can provide valuable insight on the performance of a particular scenario. The simulation of multiple scenarios and the possibility to compare key outcomes, adds further value and support to decision makers [9, 12].

*Simulation is a tool to evaluate the performance of a model, under different configurations and over long periods of real time, to reduce the chances of failure to meet specifications, to eliminate unforeseen bottlenecks, to prevent under or over-utilization of resources (including people and money), and to optimize system performance.* Bizagi Simulation follows BPSim (Business Process Simulation) standard that allows enhancement of business process models captured in BPMN to support rigorous methods of analysis [9, 18]. To start using simulation in Bizagi all is needed is a complete Process model. Bizagi Simulation comprises of four levels. Each subsequent level incorporates additional information exhibiting more complexity than the preceding one, thereby providing a detailed analysis of the processes. Levels are not interdependent, since is possible to start at any level if you hold the required process data [9, 12].

**Level 1 - Process Validation:** The first and most basic simulation level to evaluate the structure of the process diagram.

Data: It requires estimated percentage splits of sequence flows to provide a basis for routing. It also needs the value of the trigger counter contained in the Start Event shape.

Results: The outcomes show all paths activated during the execution and whether all tokens actually finished. Additionally, it evaluates how many tokens passed through each Sequence Flow, Activity and End Event [9, 12, 13].



**Level 2 – Time Analysis:** Second level of simulation to measure the end-to-end process time.

Data: Apart from the data entered in Process Validation, estimated timings (service times) of each activity and the interval time between token generation is required. This data can either be constant or samples from statistical distributions.

Results: The results show process throughput times for tokens, presented as minimum, maximum, mean and sum (total of all processing times). Similar results can be presented for individual key activities [9, 12, 13].



**Level 3 – Resource Analysis:** Predicts how the process will perform with different levels of resources. This level of detail provides a reliable estimate of how the process will perform in operation.

Data: In addition to the data entered in Time Analysis, this level includes the definition of resources (and/or roles): how many are available and where they are used. Due to the inclusion of resources, the activity times should be adjusted to represent the actual work time; delay due to unavailability of staff will be explicitly indicated.

Results: The structure of the results is similar to Time Analysis. Also, the time spent, the time spent busy or idle for each type of resource is presented. This level assumes an unlimited number of resources [9, 12, 13].



**Level 4 – Calendar Analysis:** Includes calendar information that reflects the process performance over dynamic periods of time, such as shifts, days schedules or weeks. By default Bizagi includes a chosen calendar that works 24/7. If no calendars are defined, Bizagi will assume that the defined Resources will always be available.

Data: Apart from the data entered in Resource Analysis, it includes the definition of resource calendars.

Results: The structure of the results is similar to Resource Analysis [9, 12].



## Scenarios

Bizagi Simulation allows you to create multiple scenarios for your process model, to analyze different combinations of data input and observe many possible outcomes.

Scenarios are completely independent from one another, from the definition of the scenario itself to the data included in each shape of the model [9, 13, 14].

### **What If analysis**

*What If* analysis is a powerful tool for improvement that evaluates how strategic, tactical or operational changes may impact the business. Through different scenarios business analysts are able to perform a true-to-life analysis of business processes without putting business operation at risk. Bizagi allows to easily carry out what-if analyses on processes to evaluate, understand and predict the effects of management decisions over given performance measures. *What if* analysis can be performed in any of the simulation levels and provides answers to questions like [9]:

- How would the processing time of a case decrease if the number of available resources is doubled?
- What would be the cost/benefit rate of reducing the process time in a specified activity?
- What would be the effect of altering the working shift configuration in the operational cost and service level [9]?

### **7.3 Bizagi Studio**

Bizagi Studio turns process models into running applications so that can be distributed across the organization. No coding, no hassle, just a simple web application that allows to

easily adapt business processes to the organization ever-changing world. Below are presented in high level the main features [1].

- Process IS the Application - Turn processes into a web application and adapt it instantly. No coding required and all in-line with the latest web standards
- Achieve results fast - Data layer promotes reuse, enabling the quick and cost-effective share of business objects across processes and projects
- Code-free forms – Create the user interface by drag and drop attributes into auto-generated forms.
- Flexible business rules - Manage and share business policies and rules across many projects. Make changes easily and implement instantly.
- Workload balancing and routing - Gain total control over resources. Define work to be allocated. Optimize and balance workloads through inbuilt algorithms and manage delegates and working calendars.
- Easy integration - Link Bizagi data with other IT assets present within the organization using the powerful SOA-based engine. From SAP to Documentum, SharePoint to Outlook, connect to processes without programming [1].

Bizagi Studio presents an integrated runtime environment where automation process are deployed and can be tested. As the building process takes place, this runtime environment, or Work portal, reflects in real time all the changes performed in Bizagi Studio [1].

### **7.3.1 Bizagi Studio user interface and the automation steps**

Bizagi Studio has two main views: the *Wizard* view and the *Modules* view. Every project can be automated using the Wizard view; however, some advanced features are available



only in the Modules view. In order to change between the views, locate the Home tab in Bizagi Studio's upper ribbon, and select the Modules/Wizard toggle button in the View group. The Wizard view is presented in Figure 23 [1].



Figure 23: Bizagi Studio Wizard view

Bizagi Studio provides a very powerful Process Wizard that guides through all the necessary steps to automate and execute business processes. Simply by following each of the seven steps present in the Process Wizard is possible to transform processes designed

with Bizagi Modeler into applications without the need of programming [1]. The steps of the Process Wizard are shown in Figure 24:



Figure 24: Process Wizard steps

- **Model Process** - Define the Process flow by using Bizagi Modeler.
- **Model Data** - Design a data model that organizes the cases' information that will be used in the different activities of the process.
- **Define Forms** - Design the user's interface and the information that will be displayed in the activities of the process.

- **Business Rules** - Define conditional flows and expressions to model business behaviors.
- **Performers** - Assign the users that will carry out activities in processes.
- **Integrate** - Configure connections with external systems or between processes. This step is optional.
- **Execute** - Deploy the project to Test or Production environment.

### 7.3.2 Process modelling

The first step to create Bizagi solutions is to design the process model. The preferred approach to design process models is to use Bizagi Modeler as described in section 5.2. After the model is completed in Bizagi Modeler it can be exported in Bizagi Studio and continue with the other automation steps. There is also the possibility to use the process modeler included in Bizagi Studio to design models. This second methods does not provide the capability to documenting the process and that is the reason the first approach is the one preferred and the one followed during our work [1, 2].

### 7.3.3 Data modelling

The second step of the Bizagi Process Wizard is to define the data that the Process requires for its execution. Bizagi provides a friendly wizard to build a structured data model with defined entities and their corresponding attributes. Business data can be structured in a graphical and logical way. Data is created only once and used throughout the process and can be reused in all processes created in the same project without restrictions. It is very important that the data in projects is correctly organized to achieve the Processes goals. To

provide an organized and coherent structure Bizagi provides four types for Entities and four types of Relationships with which we can build the data model [1, 3].

## Entities

- Entities are real or abstract objects (people, places, events, and so on) that can be uniquely identified and are of interest to the business about which information is stored in the system and can be usually thought of as nouns. For example; *a Customer, a City, a Company, an Invoice, a Car.*
- Entities have Attributes. These are the properties that describe each entity. For example *a customer has a name, a social security number a gender and an age.*
- Entities should always have one attribute or set of attributes by which it can be uniquely identified. For example, *a Customer has a social security number that is unique.* No other record can hold the same identifier value.
- Bizagi generates consecutive row numbers automatically which uniquely identify each record in an entity. This identifier is called a surrogate key. It has no connection to the data attributes in the row but simply makes the whole row unique [1, 3]. In Figure 25 is presented an example of an entity.

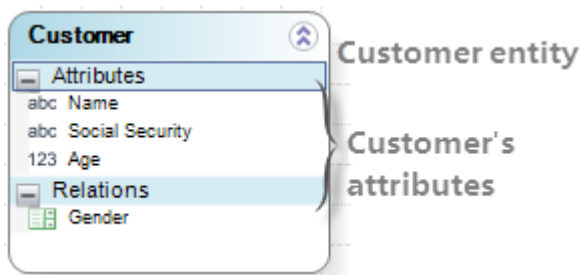


Figure 25: Example entity in Bizagi

Every process in Bizagi has its main Process Entity. This entity provides the starting point to access the rest of the process data, that is, it is the principal entity through which a user accesses the rest of the data model entities. It is the first entity that is created and it has a double line that will distinguish it from the rest of the entities [1].

## Relationships

- Relationships capture how entities relate to one another and can be thought of as verbs. For example: a Customer *owns* a car. A company *is located* in a city. A customer *has* a gender.
- With every relationship a foreign key is automatically created. A foreign key is an attribute that is the primary key in another entity.

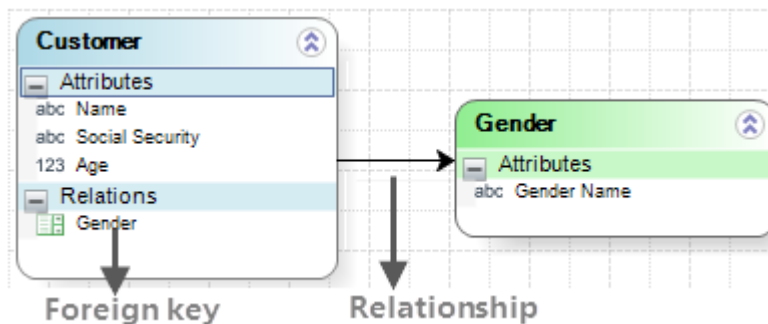


Figure 26: Example relationship in Bizagi

### 7.3.3.1 Entity types

Bizagi offers four types of entities that help each process to have an easily accessible organized data model [1, 3].

#### Master entities

Master entities are business entities that store information that is directly and exclusively related to each process case. The information stored is stored as a process flows. That is, the data is stored according to what the end user enters in the Work Portal. The *Process Entity* is the main *Master Entity* and connects the related Master entities in the Data Model. There can be included as many Master entities as the project requires. In each entity it can be included a maximum of 85 attributes (even though it is not recommended nor usual to have more than 30 attributes). Exceeding the proposed maximum amount may compromise the performance of the project. In the data model diagram, Master entities are shown in blue color [1, 3].

### **Parameter entities**

These entities store predefined values, or parameter values, that are independent from the processes' execution. During a case their values cannot be modified or written to. For example, the entity Gender contains the values Male and Female [1, 3].

It is possible to relate the parameter entities information to a case establishing a relationship between a master entity to a parameter entity. For example the entity *Customer* will have a relationship with the entity *Gender*. This way the entity Customer will store the value of the gender entity that is selected for each specific case. There can be included as many parameter entities as the project requires. In each entity is possible to include a maximum of 85 attributes. Exceeding the proposed maximum amount may compromise the performance of the project. In the data model diagram, Parameter entities are shown in green color [1].

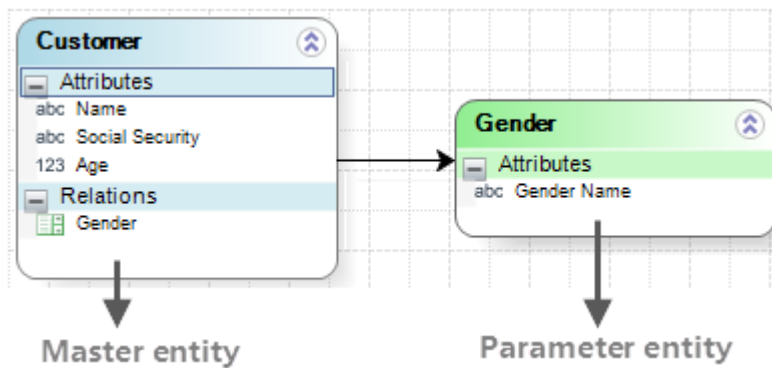


Figure 27: Example of Master and Parameter entity

### System entities

These entities belong to the Bizagi's internal data model. This group of entities includes information concerning the end user, areas, locations, roles, skills among other. System entities are created by default for each project. It is not possible to create additional system entities nor add or modify its attributes. However, is possible to create relationships from other entities to include System Entity information in the data model [1].

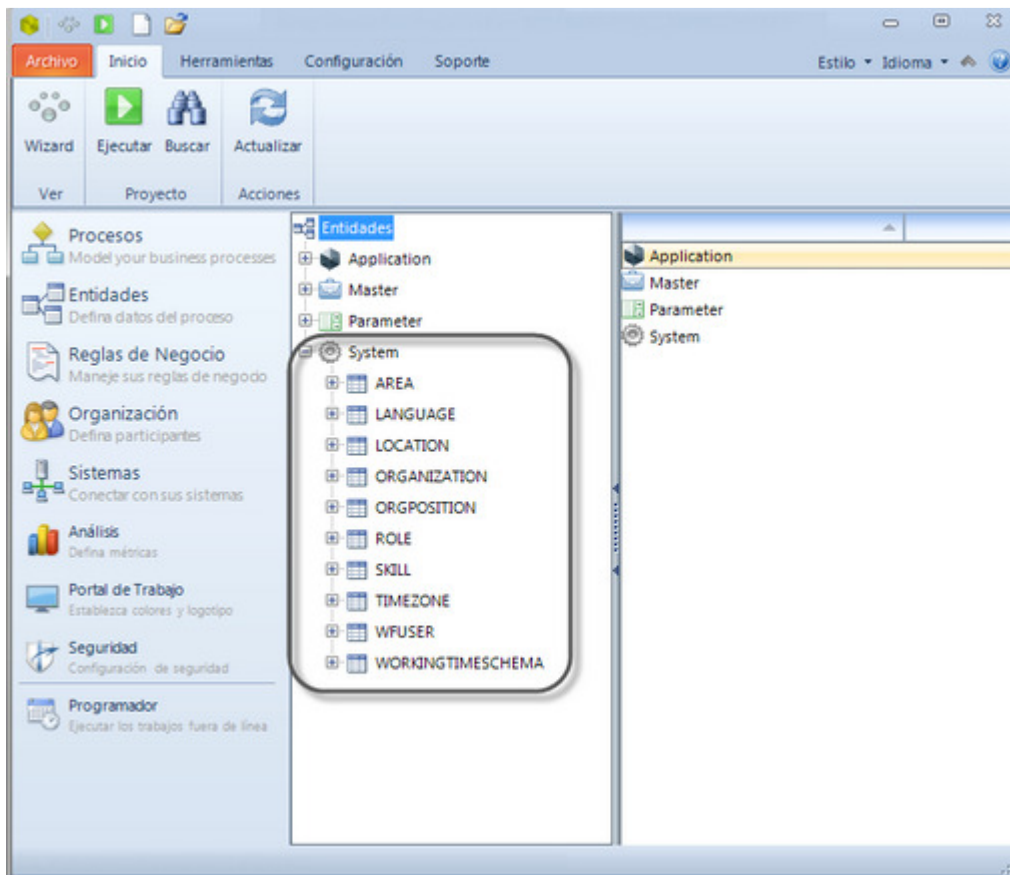


Figure 28: System entities in Bizagi

### Application entity

Application entities centralize the information of each application. These entities manage the entire data model for the whole project. Application entities are generated automatically when an application is created, to allow for structural organization of processes, and it is named after the application. Users will not be allowed to access or delete these entities [1].

#### 7.3.3.2 Relationship types



Bizagi offers four types of relationships between entities, providing greater flexibility in your data model.

### **Related Attribute relationship**

This is the most common relationship in which one instance of an entity is associated with one instance of another entity. What distinguished this relationship is the order in which it is created; mainly because it creates an attribute in one entity in reference to another but NOT both ways. It is commonly used for relationships between a master entity and a parameter entity for drop-down lists (or combos) and between master entities. This type of relationship is automatically created through the Data Model Wizard, when choosing the Entity type attribute [1].

For example a Customer has one gender, thus there is a Related Attribute between Customer and Gender entities. However a gender will be assigned to many customers. The Related Attribute relationship creates an attribute in the Customer entity, shown under Relations, to reference the Gender (not both ways). You cannot reference a particular Customer from a Gender. The relationship is modeled as a single arrow line.

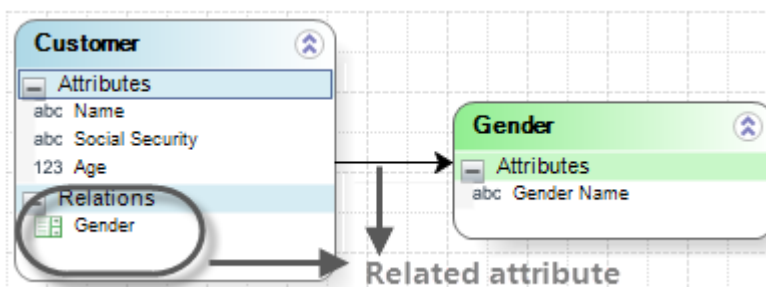


Figure 29: Related Attribute relationship

### **One-to-one relationship**

One-to-one relationships are established when there is a single correspondence between two entities. That is, each record from entity A is associated with a single record from entity B and vice versa. For example imagine there are two entities, Employee and Computer. There is a single to single relationship between the two entities because each employee is associated with just one computer and one computer has been assigned to only one employee. When this type of relationship is created, Bizagi will automatically create the attributes (foreign key) that relate the entities to each other. The one-to-one relationship creates a foreign key attribute in each entity, shown under the respective Relations nodes, to reference the other entity (both ways). The relationship is modeled as a double arrow line [1].

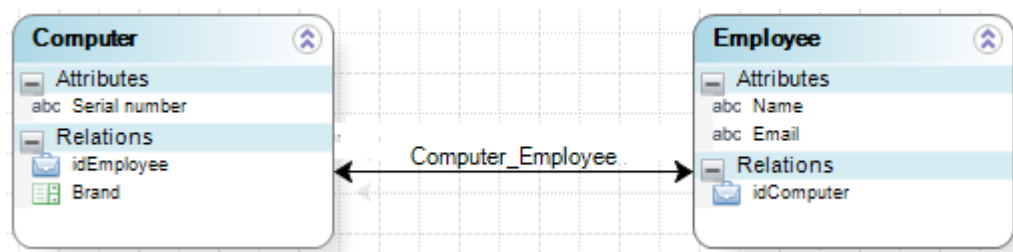


Figure 30: One-to-one relationship

### Collection or one-to-many relationship

One to many relationships are established when one instance of an entity (entity A) can be associated with zero, one or many instances of another entity (entity B). However for one instance of entity B there is only one instance of entity A. In Bizagi this relationship is called a Collection [1].

For example, imagine there are two entities: Customer and Requested Products. The Customer (entity A) can have many Requested Products (entity B) but those products that

where requested can only belong to one customer. These type of relationship is automatically created through the Data Model wizard, when selecting the Collection type attribute. A collection is modeled as a single arrow line with an asterisk (\*). Note that no attributes are created and shown under Relations of the Customer entity for this relationship [1].

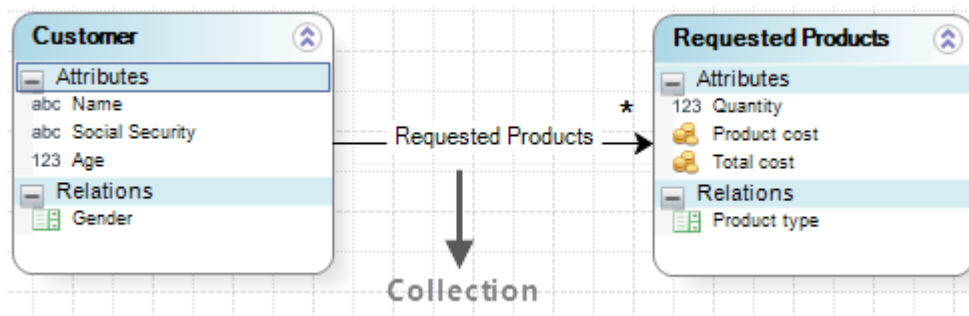


Figure 31: One-to-many relationship

### Multiple-to-Multiple relationship

Multiple-to-multiple relationships are established when one instance of an entity (entity A) is associated with one, zero or many instances of another entity (entity B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, in a Loan Request Process, a request may have several products (personal loan, credit card) and several collaterals, or guarantees to cover the products (co-borrower, mortgage). Each product may have many associated guarantees. Thus, there is a multiple to multiple relationship between Guarantees and Products: A product can be covered by several guarantees of the request, and a guarantee can cover several products of the request. A many-to-many relationship is modeled as a double arrow line with an asterisk (\*) at each arrow end [1].

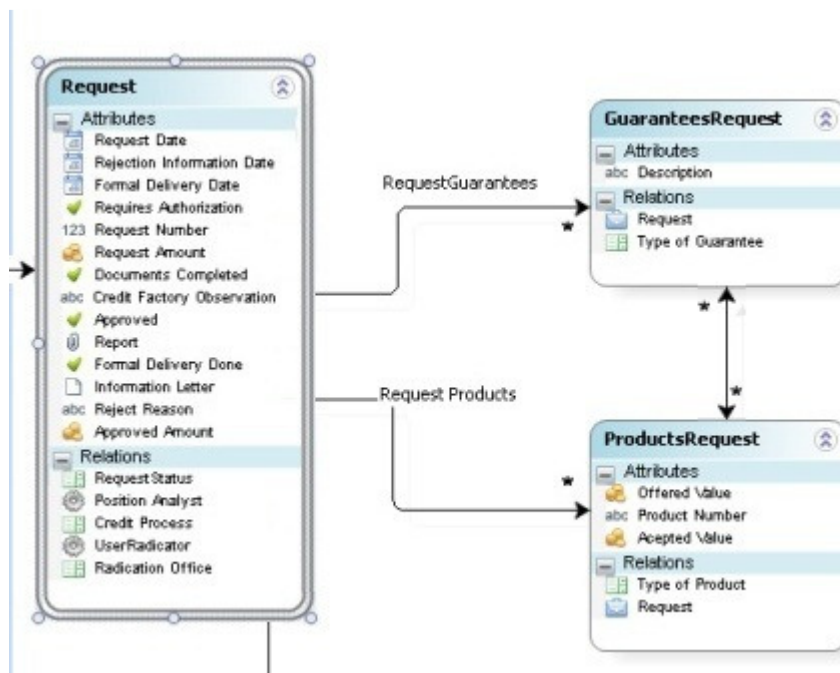


Figure 32: Many-to-many relationship

### 7.3.3.3 Attribute types

Bizagi offers a set of attribute types. They are separated in four categories to help to quickly find the one needed [1]:

- **Basic types:** the most common types used. They are on the top of the list for easy access.
- **More types:** Additional advanced types.
- **Entity:** Gives access to all Application, Master, Parameter and System entities of the model. Accessing entities through this category creates a Related Attribute relationship with the target entity selected. Is possible to create any type of entity needed if it has not been created; the relationship will be created automatically.

- **Collection:** Accessing entities through this category creates a One-to-many relationship with the child (target) entity selected. Is possible to create collections with Master and Application entities on the data model. If needed is possible to create a new entity; the relationship will be created automatically.

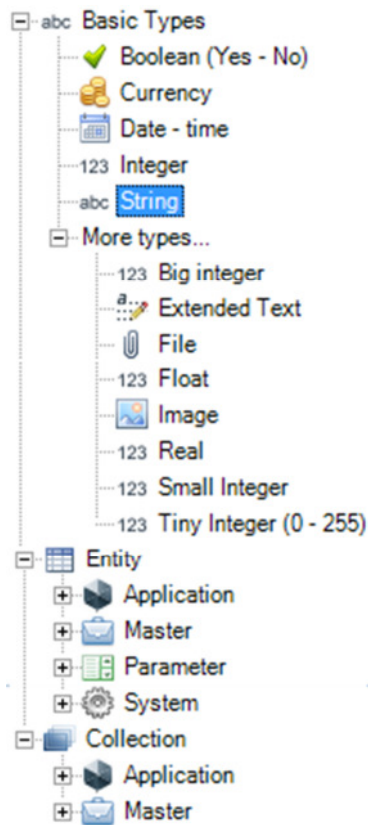


Figure 33: Attribute types in Bizagi

#### 7.3.3.4 Example Data Model creation in Bizagi

The second step of the Bizagi Process Wizard is to define the data that the process requires for its execution. Let's use the use the Purchase Request Process as modelled in section 5.2.1 to explain the construction of the data model [1].

1. Select the Model Data option in the second step of the Process Wizard.



Figure 34: Data Modelling, second step of the process wizard in Bizagi

2. A new window will display a drop down list to select an entity. Enter the name of the main Process Entity or select an existing entity from the drop-down list.  
Remember that the Process Entity is the principle entity of your process, through which a user accesses the rest of the data model entities. Click OK.

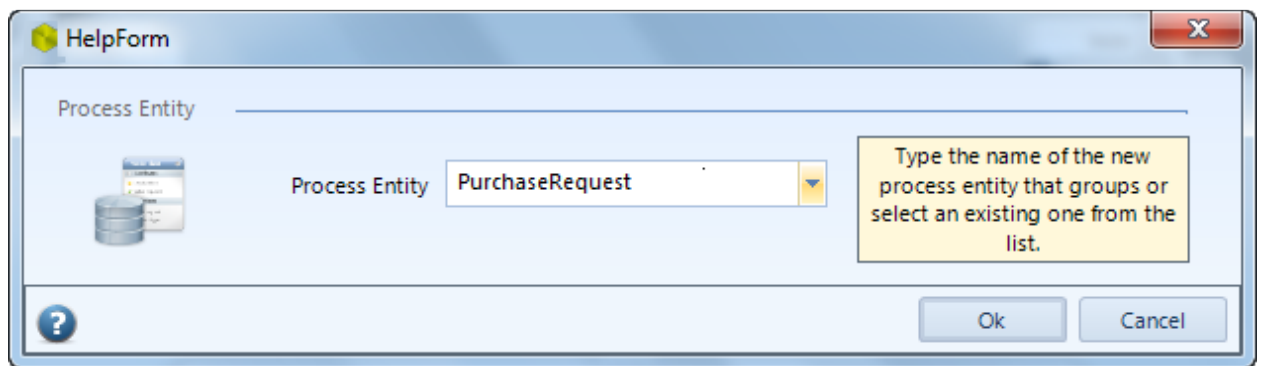


Figure 35: Main process entity creation for the Purchase Requisition process

3. Right click the Process Entity and select Edit Attributes list, to start including attributes.

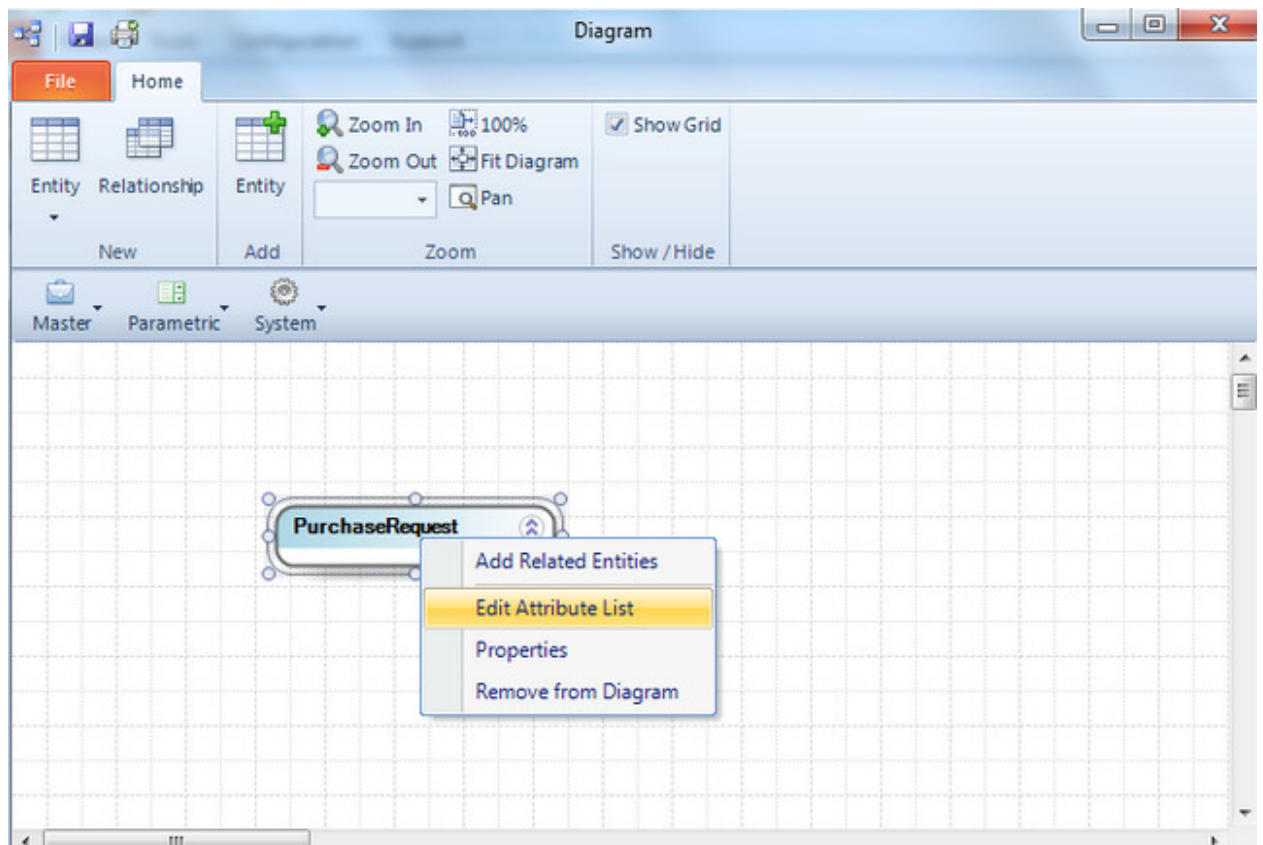


Figure 36: Adding attributed to entities in Bizagi

4. The Entity Wizard will launch allowing you to include all the attributes you need.

Click the Add button. A new record will be added to the table where the first field holds the attribute's display name. The Display Name is the name that will be displayed in the Work Portal user interface. The Name attribute is the unique identifier for internal use only and is automatically created. The last column specifies the attribute's data type. For example the delivery date of a purchase should be a date (i.e., the Date-timedata type). The delivery address should be a string (i.e., String data type).

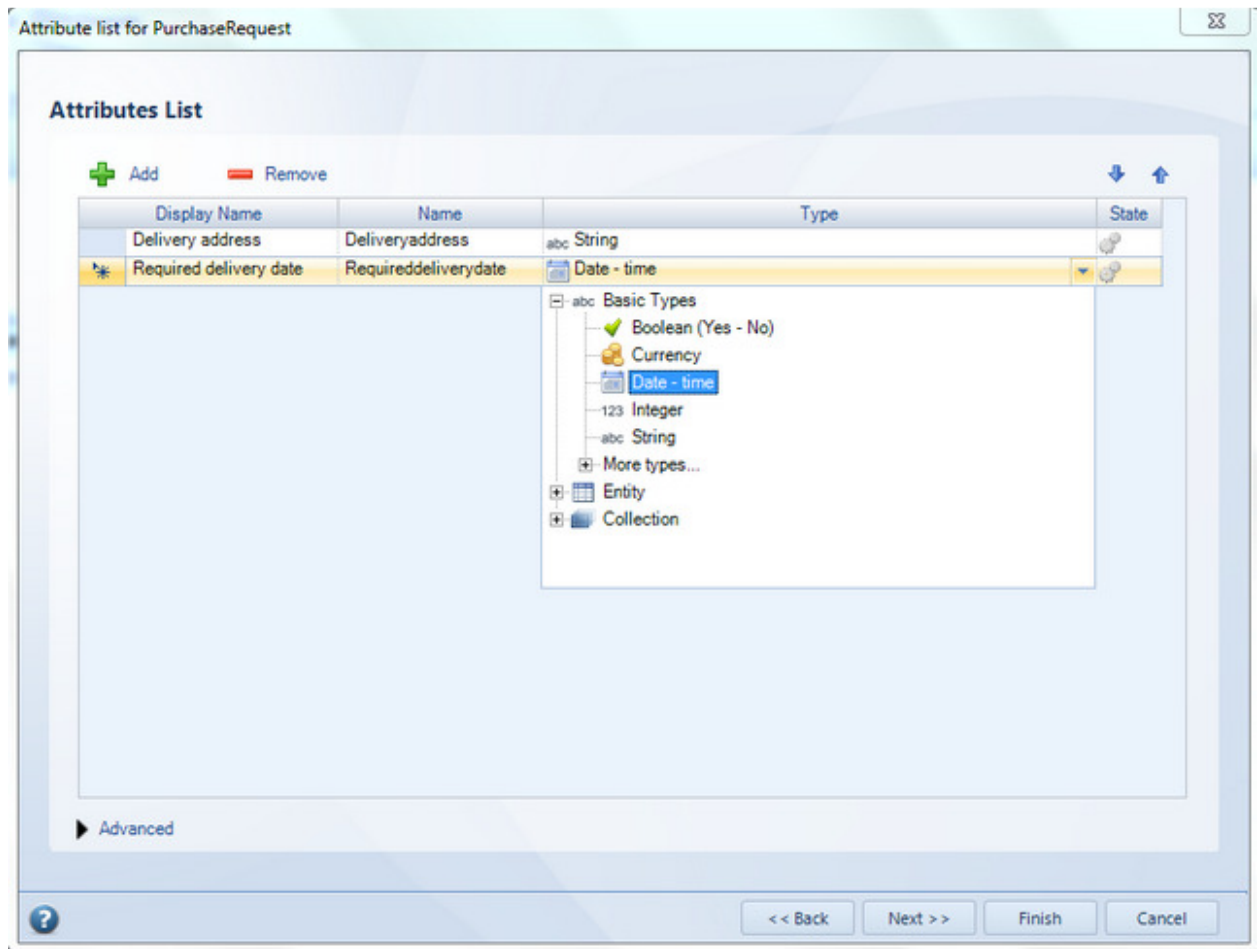




Figure 37: Attribute creation in Bizagi

5. You can also create attributes that relate Master and Parameter Entities. In this example we will add a relationship to the Parameter Entity City. In the Display Name you include the name of the Relationship. Type the name of the relationship in the Display Name field, and select the child entity. To do so, first expand the Entity element (by clicking the plus sign aside) and then the Parameter element. Select New Entity or choose an existing entity from the list presented.

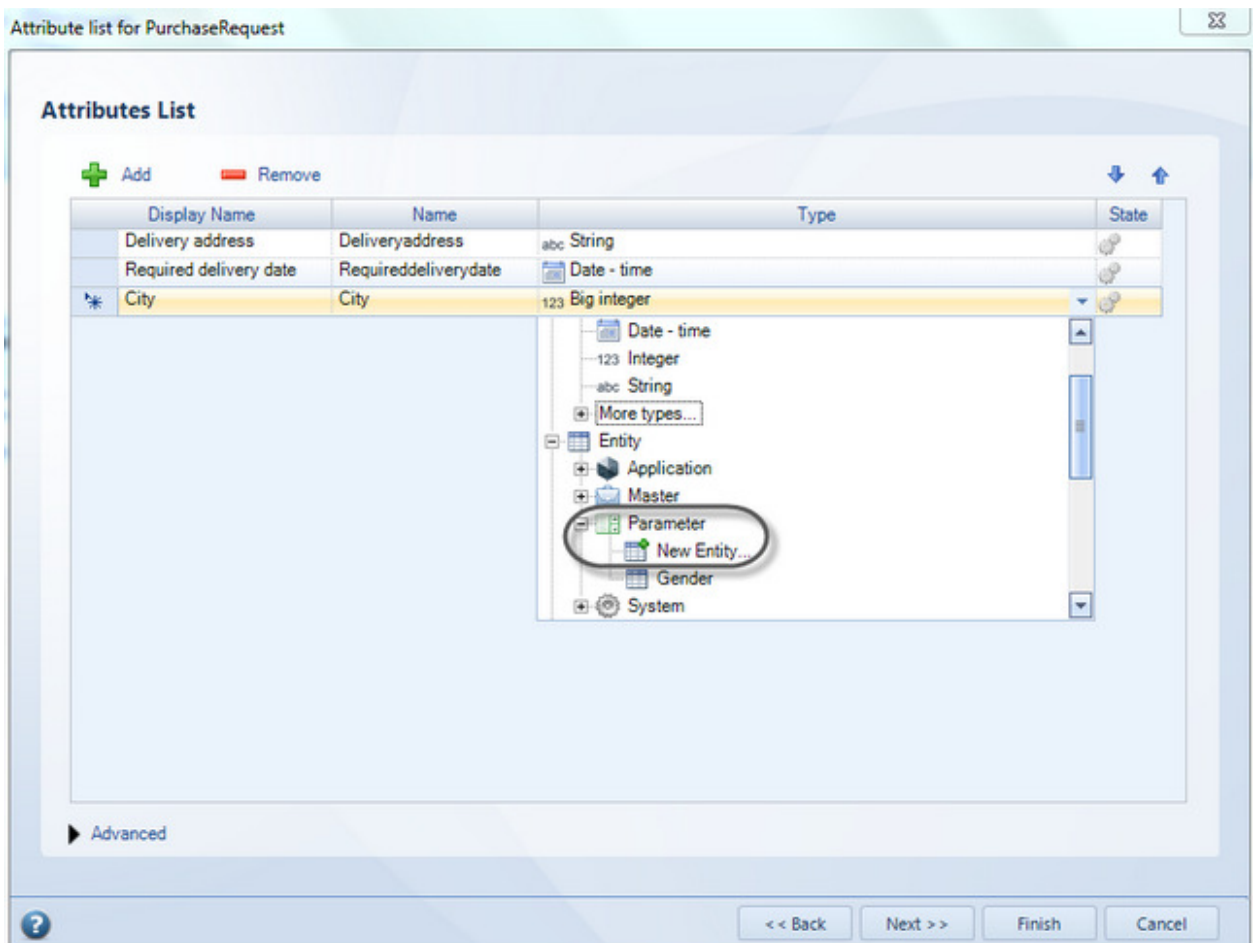


Figure 38: Example of attributes that relate Master or Parameter entities in Bizagi

6. The Attribute List window will appear to enter all the attributes of the new Parameter Entity.

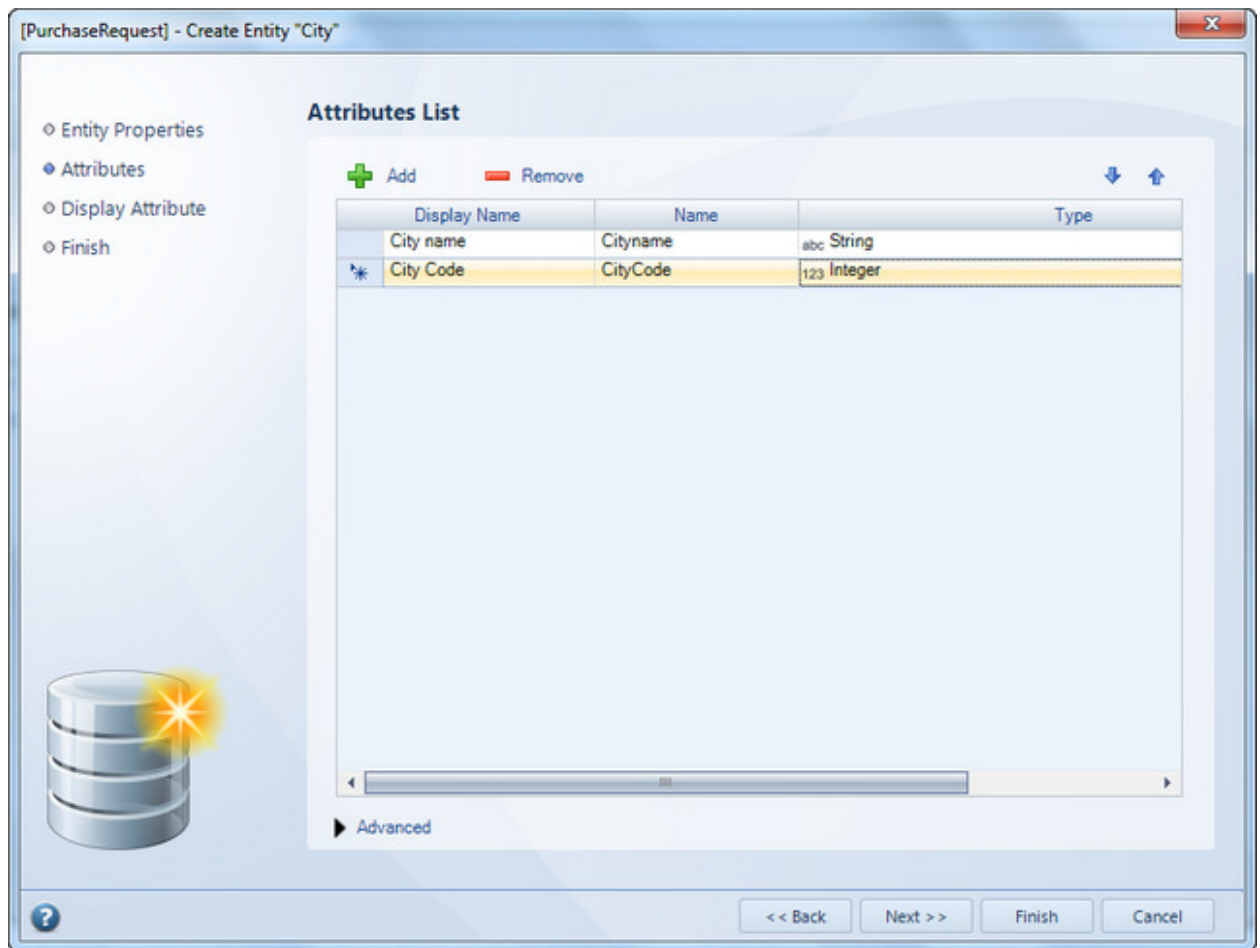


Figure 39: Attribute list window in Bizagi

7. In the last step of adding the Parameter Entities, you must define the Display Attribute and a parent entity (click theAdvanced/Hide toggle option to see this field) if there is one associated. The Display Attribute is the visible attribute created for the entity that will be shown in the Work Portal when the entity is referred to. This

field is required but can be selected from the User Interface Modeler. Once complete, click Finish. The system returns to the previously displayed Attribute List window of the Entity Wizard, where you can continue to add other required attributes. When all attributes have been added, click Finish.

#### **7.3.4 Creating the user interface**

End users interact with the automated processes through a web designed portal that executes with any web browser. Users access this interactive process portal called Work Portal, where they have access to all their cases with activities pending. Each pending activity is represented by a user interface, known as a Form in Bizagi that displays its data. The third step of the Process Wizard, Define Forms, manages all user interfaces for human activities. The Forms Designer provides an intuitive and user-friendly structure to drag and drop the data fields (known as Controls in Bizagi) onto a form and arrange them in any way the process calls for, without the need of programming. The Forms Designer has a what-you-see-is-what-you-get (WYSIWYG) approach to it, meaning that is possible to build a Form and know exactly how it will display to end users. Additionally there is the possibility to define complex validations and perform actions on the information to ensure the content entered by end users is correct and complete [1, 4].

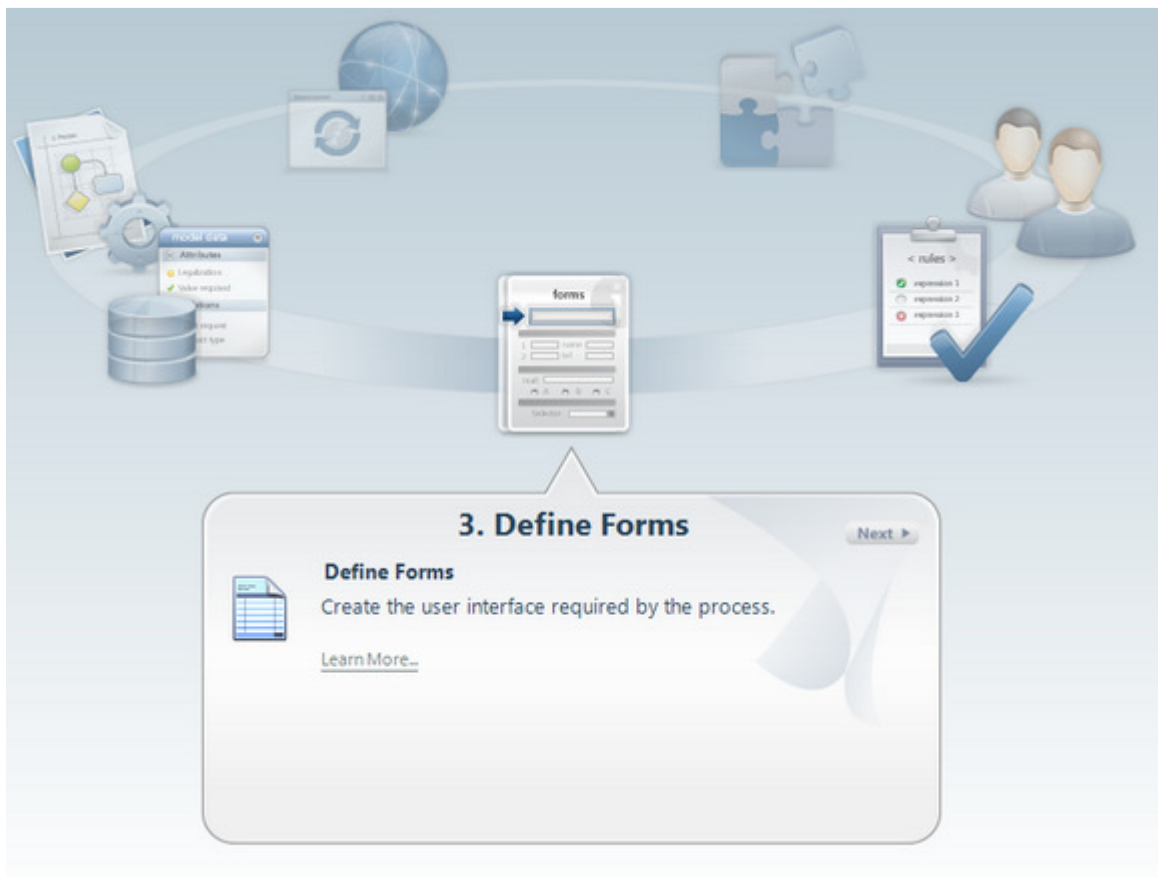


Figure 40: Define forms, third step of the process wizard

#### 7.3.4.1 Form creation

Forms are built using the Data Model defined in the previous step, Data Modeling, of the Process Wizard.

The context entity of the process is the starting point to access the data model, and therefore to each attribute that will constitute the Form. To create a Form is necessary to go to the third step of the Process Wizard and click on Define Forms to display the workflow of the Process. All human tasks that require a Form will be enabled. All other elements in the

process will be disabled as they do not receive any user input. All activities that still need a Form associated will be highlighted with an exclamation mark as shown in the image below [1, 4].

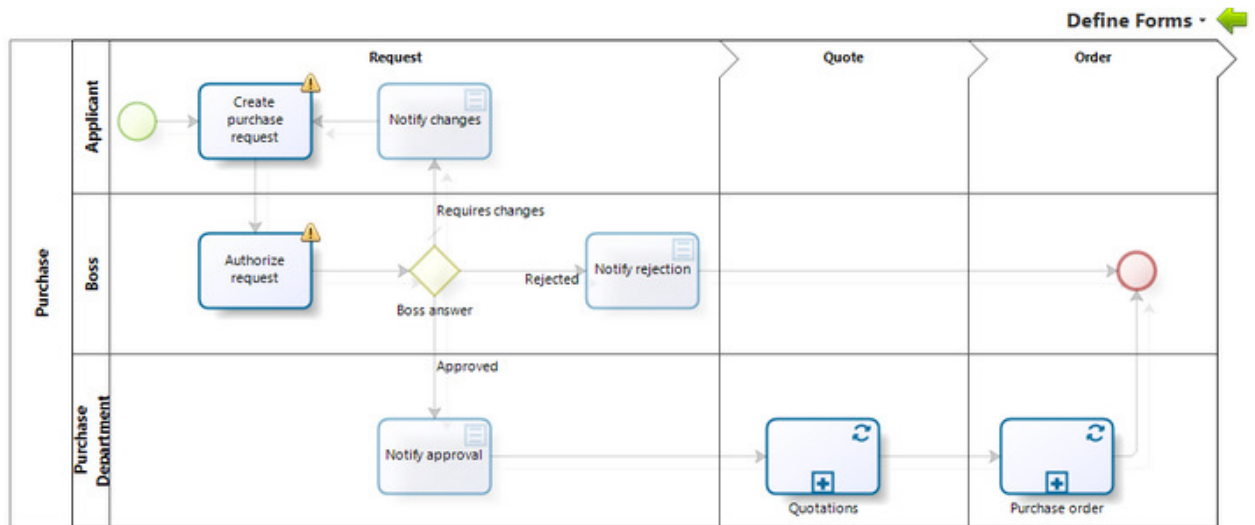


Figure 41: Third step of the process wizard, define forms for the Purchase Request process

Imagine a Purchase Request where your employees request goods and services. The process model is displayed in the image above. The first activity of the process is where the Requester enters all the required request information. To create this first Form we need to list what data must be displayed for this user interaction [1, 4].

- Request information containing: *the request date, the name of the requester and a purchase justification*
- The items requested: *table with the goods or services requested*
- Delivery information: *a delivery address, a delivery date and a delivery city.*

Let consider that to have the following Data Model displayed in Figure 42 in order to create the forms [1, 4].

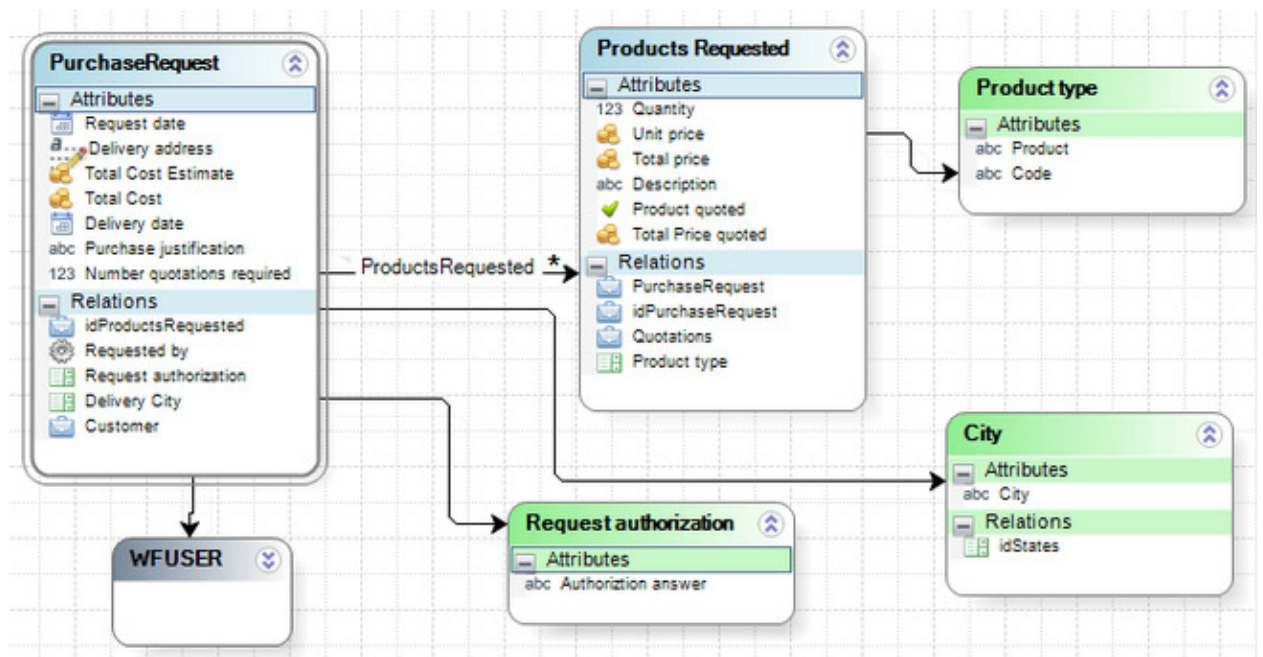


Figure 42: Data Model for the Purchase request process

1. Go to the third step of the Process Wizard. When the process model displays, click the Task to create or edit its Form. The Forms Designer will display. If there is no Form previously built the Forms Designer will show a blank display area.

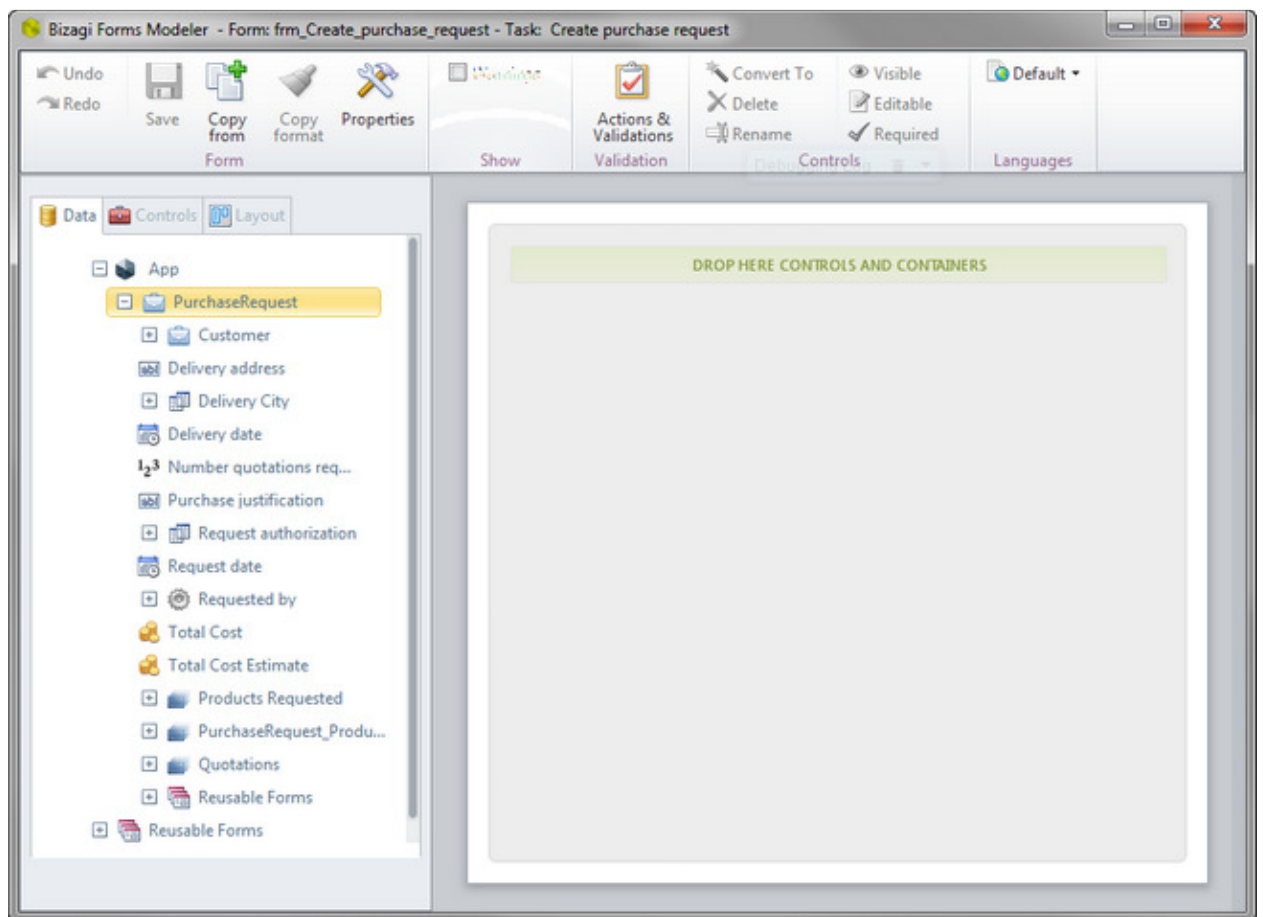


Figure 43: Forms designer

2. Let's arrange the data in three containers onto the Form. Go to the Controls tab on the left pane, select the Group Control in Containers and drag and drop three Groups. Notice that when you drag an item from the left pane, the DROP HERE section will highlight as you hover the mouse cursor over the display area. This helps to show where the current Control will be included. You can also double-click on the items to automatically included them on the bottom of the Form.

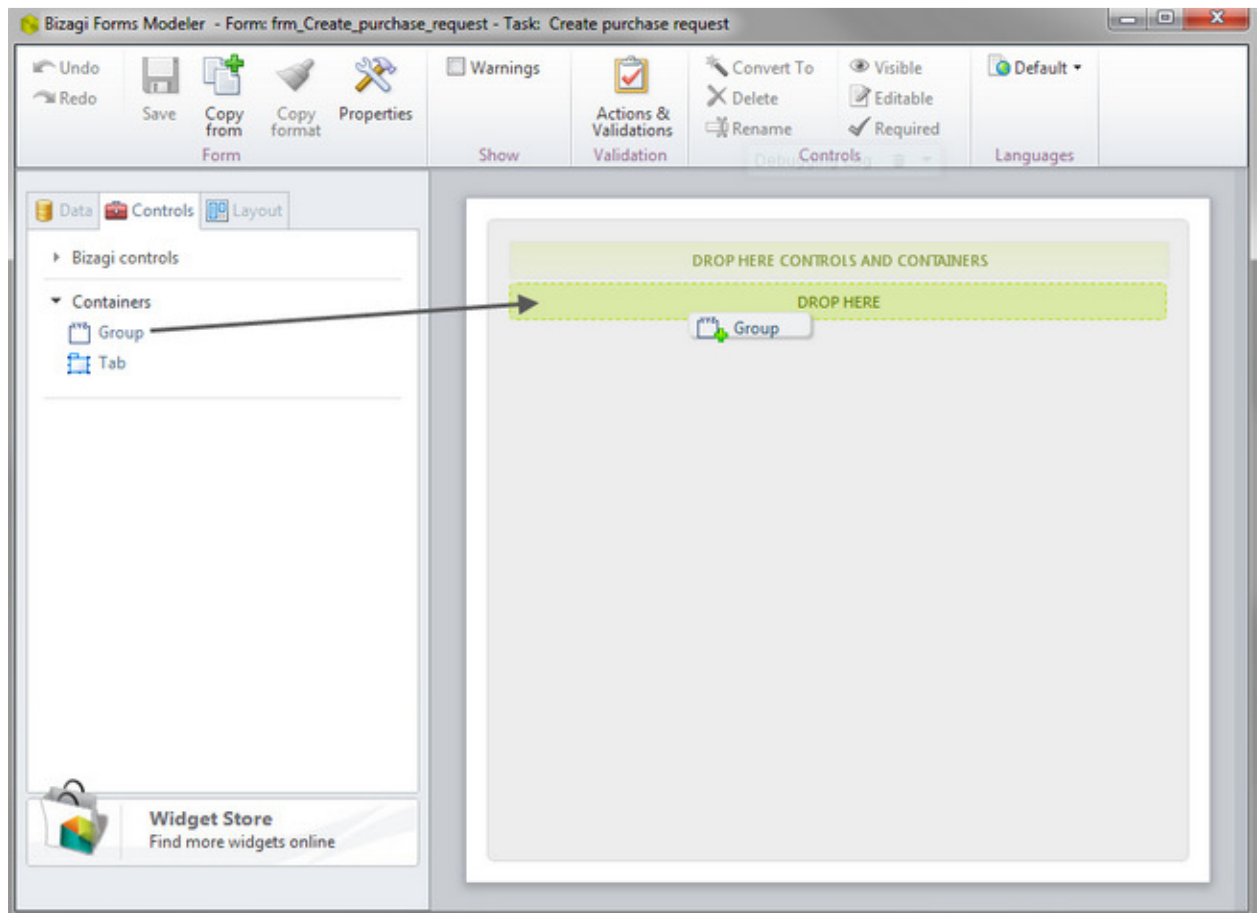


Figure 44: Forms designer, adding containers in the Form

3. Double-click on each group control to name it. Let's name the groups **Request**

**information, Products requested and Delivery information** respectively [1, 4].



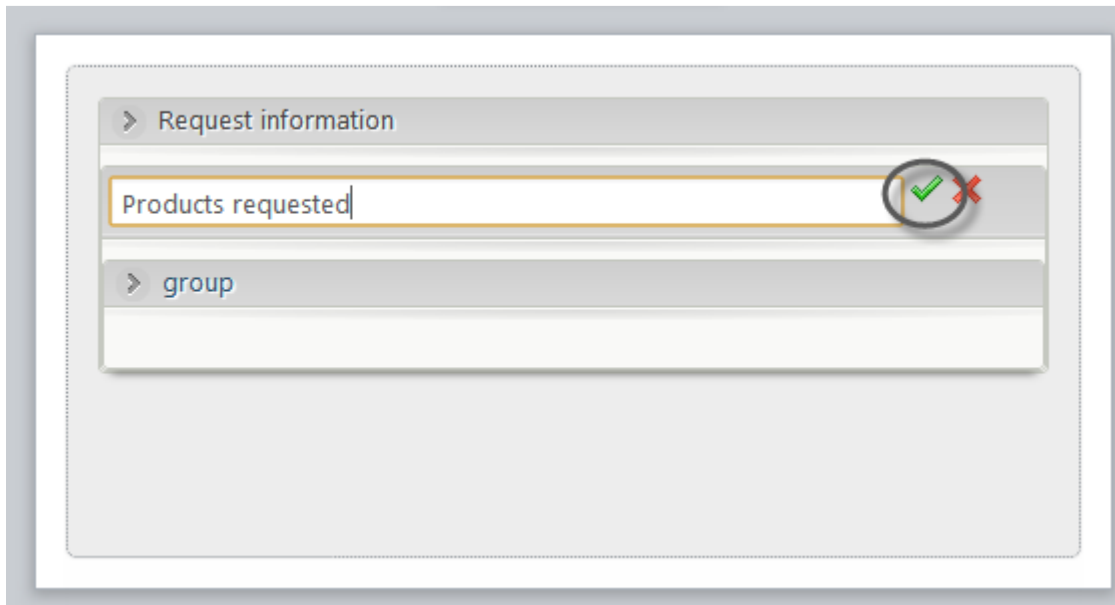


Figure 45: Naming group controls

4. Return to the Data tab. To include attributes from the data model just drag and drop them to the desired position. As soon as an attribute is added to the Form it is interpreted by Bizagi and becomes a Control. Each Control has a data type: text, Date, Combo (drop-down list), Yes/No, Number, etc, based on the attribute type. For the first Group drag and drop the attributes: *Request date*, *Requested by* and *Purchase justification*. Make sure the attributes included are located in the first Group. Drop them when the DROP HERE section of the first group is highlighted. Notice that *Request date* is included as a Date Control, *Requested by* is included as a Combo Control (or drop-down list) and *Purchase justification* is included as a Text Box Control. Bizagi automatically assigns the data type to the Control associated with the selected attribute [1, 4].

5. Drag and drop the Products Requested collection onto the second group. A collection will be displayed as a Table Control. Each column (field) to be displayed in the Table must be defined [1, 4].

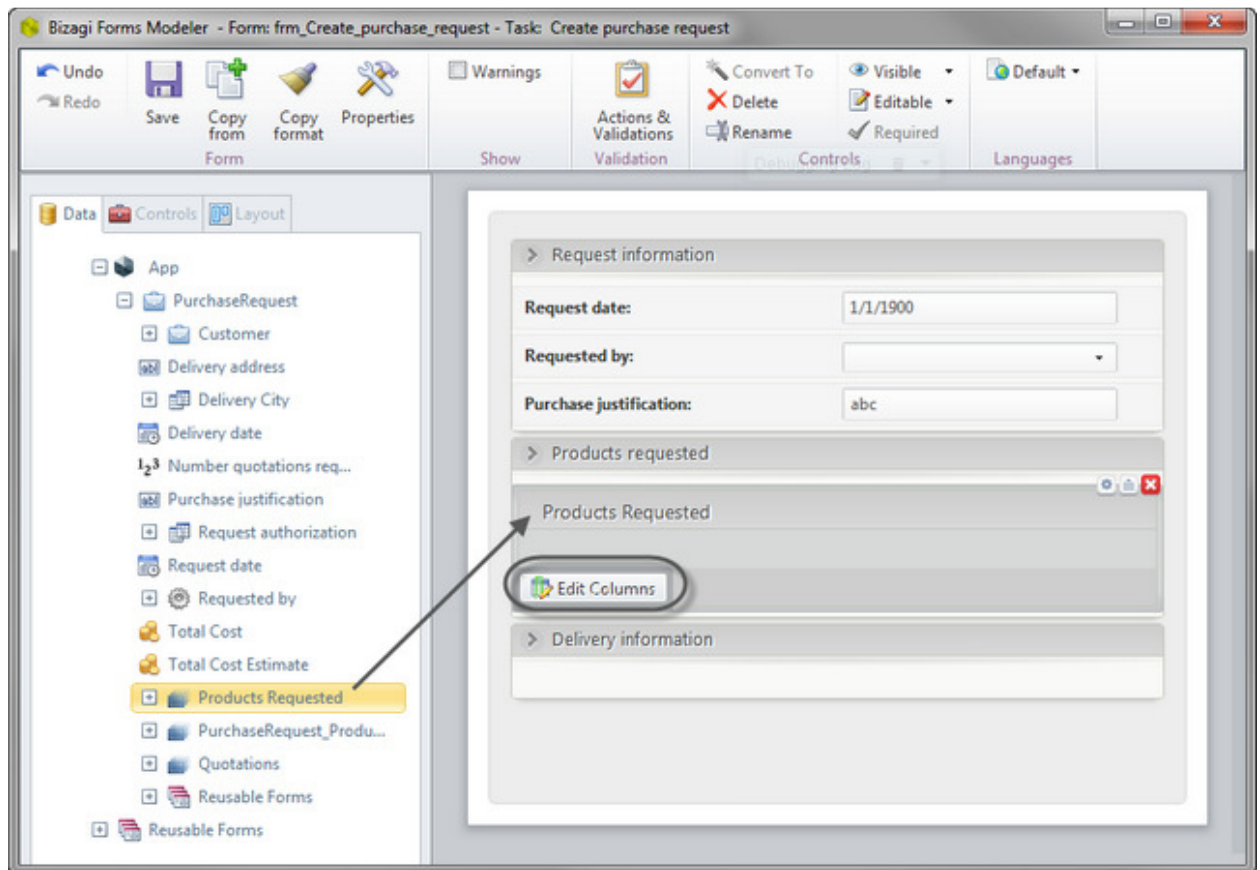


Figure 46: Define forms, progressing with form creation

6. Click on Edit Columns in the Table Control to drag and drop the columns that we need to display. You can also double-click the attributes to automatically include the associated Control in the Columns editor. Click OK when done [1].

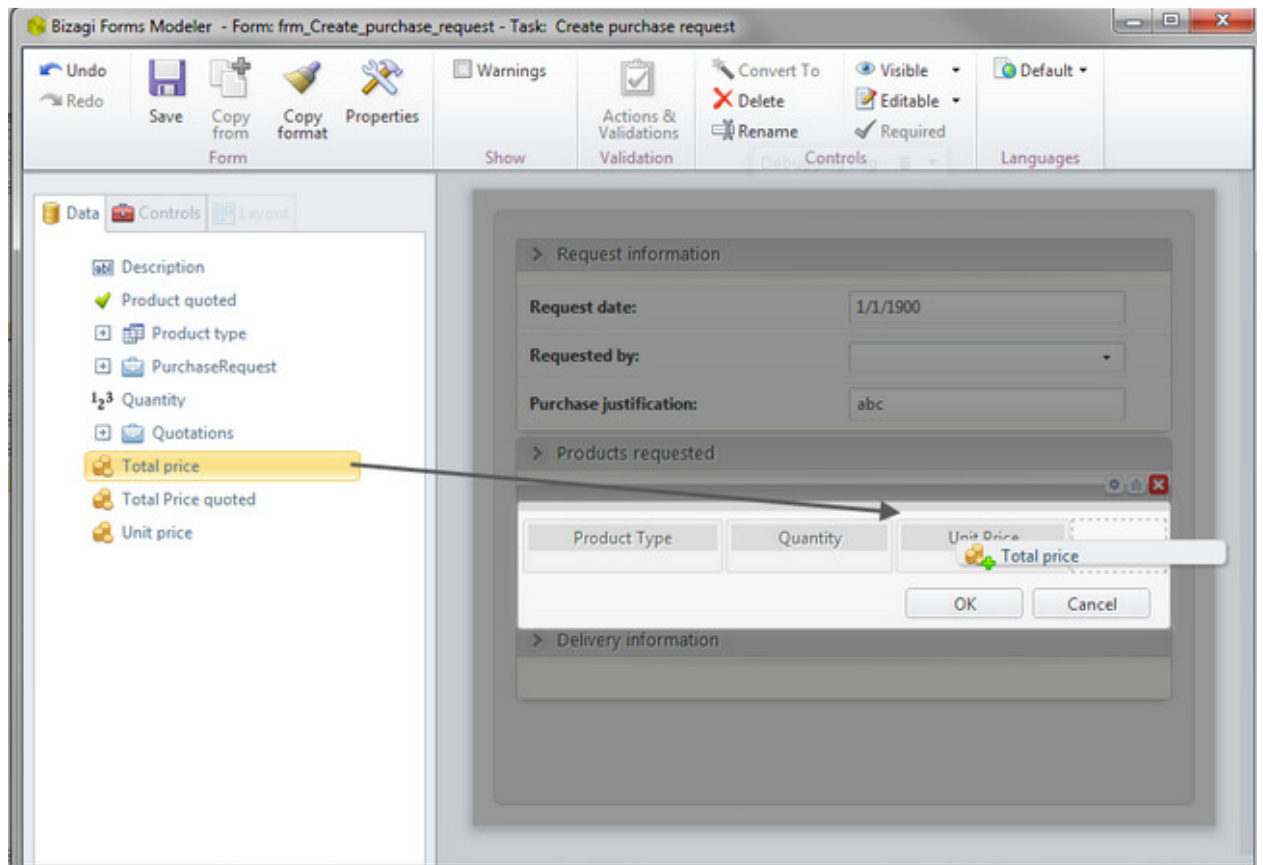


Figure 47: Define forms, editing columns in the table control

7. Drag and drop the Controls for the last Group: *Delivery address*, *Delivery city* and *Delivery Date* [1].

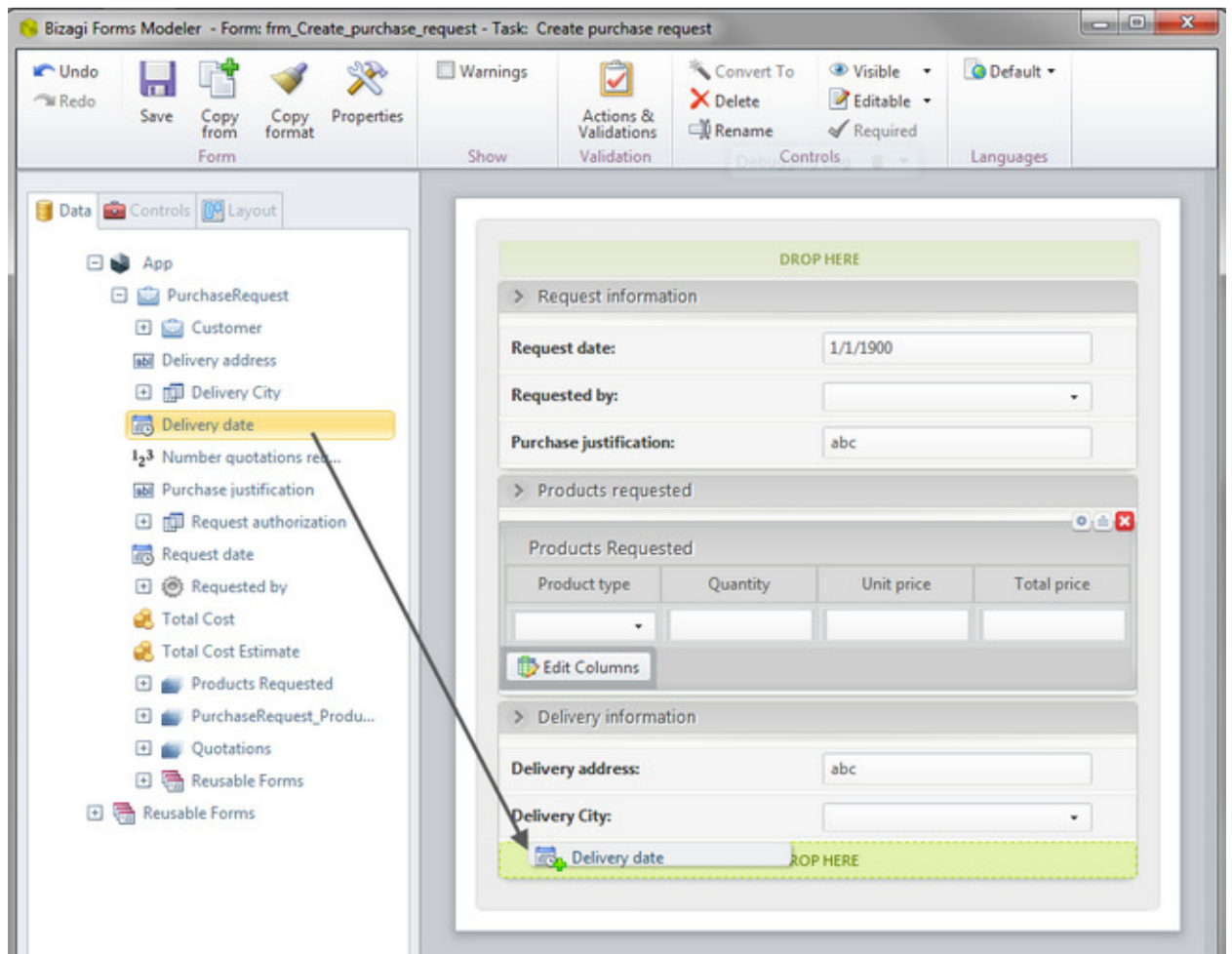


Figure 48: Define forms, completed form with all the controls

8. The first two Controls (Request date and Requested by) should be read-only. These Controls will be populated using an Expression, the data will be pre-filled when the user enters the activity in the Work Portal. Click on any Control. Its properties will be displayed on the left panel. You can also access the Control's properties by clicking the Gear icon alongside, at the top right corner of the Control [1].

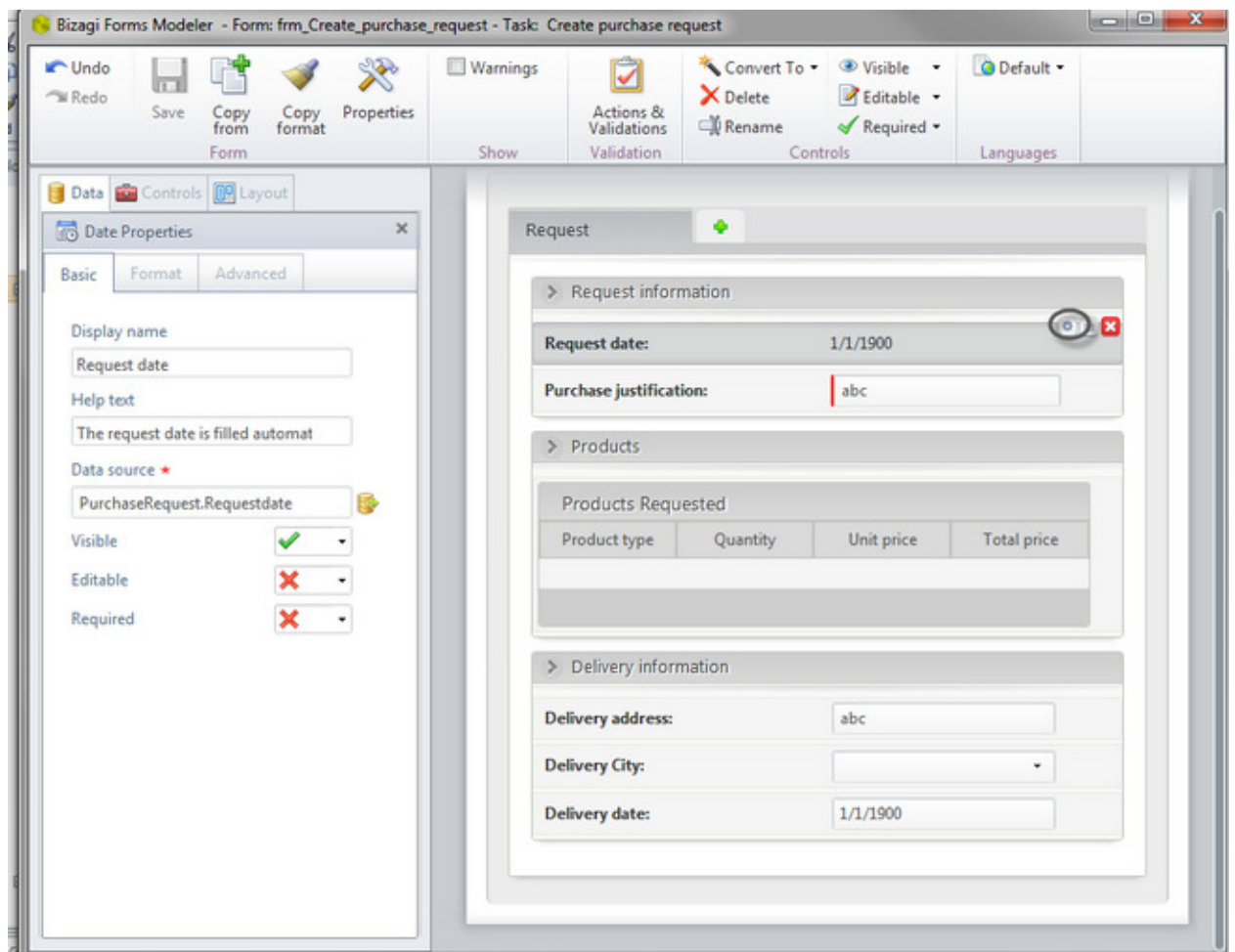


Figure 49: Define forms, properties of the control

9. Change the value of the Editable property. To make the Control read-only (not editable) select the Cross icon (X). Notice the appearance of the Control changes, it is no longer highlighted as mandatory (indicated by the red vertical bar) and has no editable area (the red bar is removed). Repeat the procedure for the Requested by Control.

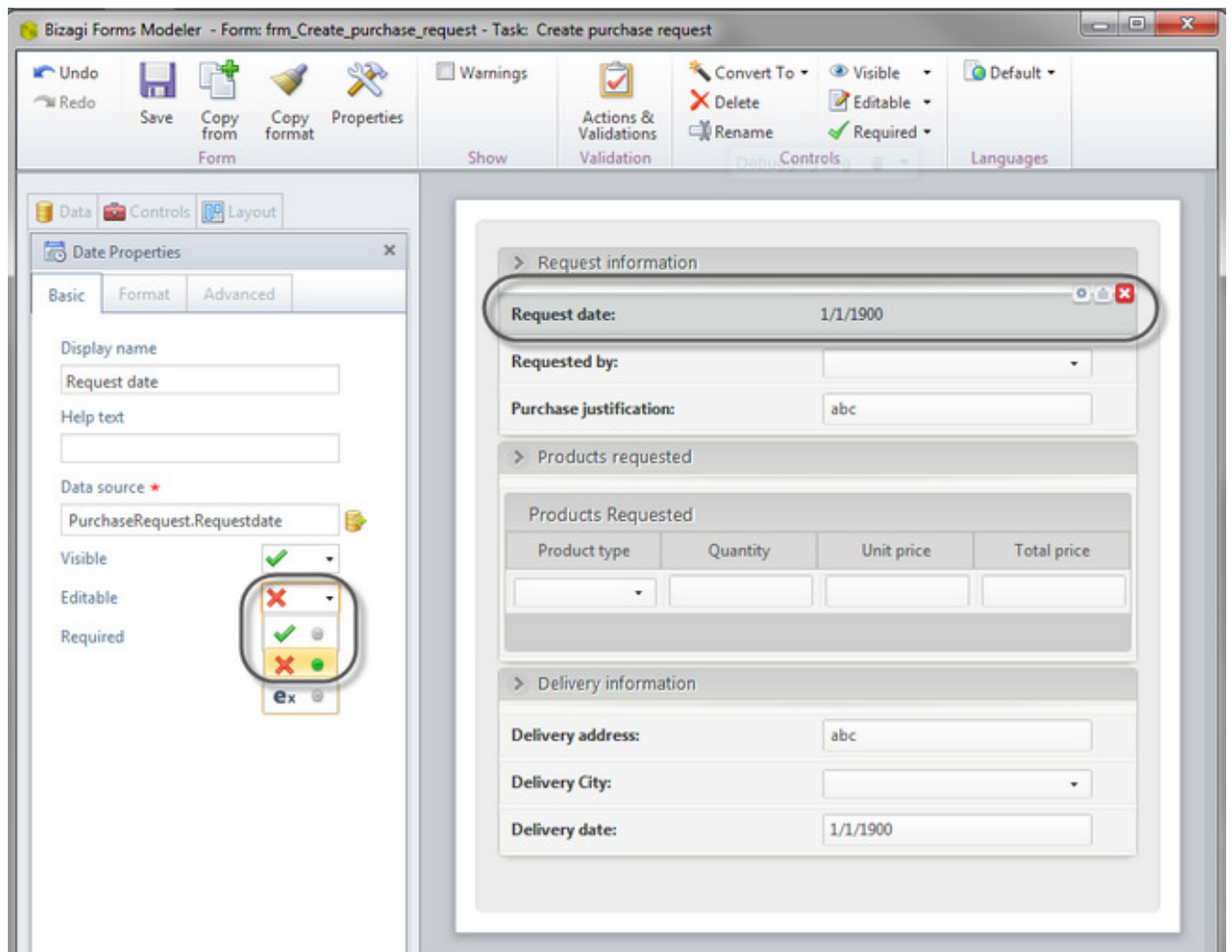


Figure 50: Define forms, changing properties for the control Request date.

10. When the form is complete, click Save [1].

In the above section is described in detail how is possible to create and design the user interface for an activity that requires human interaction. The above process can be repeated for all this kind of activities that make up the process.

### 7.3.5 Defining Business rules

Business Processes are governed by business rules that ensure their proper execution according to the strategies, objectives and philosophy of organizations; Business rules establish the procedures that must be executed and the conditions that must be evaluated and controlled in the Process flow. A process automation initiative offers organizations the possibility of yielding the evaluation and execution of these rules to Bizagi, letting people focus in the real important things, reducing errors, time and providing agility to processes. Bizagi offers a graphical environment where is possible to define and manage business rules easy and intuitively; with the powerful Business Rules Engine is possible to design from the simplest to the most complex business rules [1, 5].

In Bizagi is possible to define business rules for:

- *Routing the Process*: Controlling the sequence flow to define which path a process must follow according to specific business conditions. (e.g *If a travel request was approved, proceed with the flight booking , otherwise, notify the rejection*).
- *Performing actions in Activities*: Executing the necessary procedures when performing a task such as validations and calculations (e.g when an employee reports the expenses made in a business trip, the total value of expenses must be automatically calculated).
- *Managing the user interface*: Making form's controls visible, required or read only to avoid errors in the information entered into Processes. (e.g *If a travel request is rejected, the rejection comments control must be visible and required, otherwise they must be hidden*).

- *Allocating users:* Defining the conditions that users must comply in order to be allocated to a task. (e.g If a claim is related to Invoicing it must be assisted by a customer service agent specialized in invoicing claims) [1, 5].

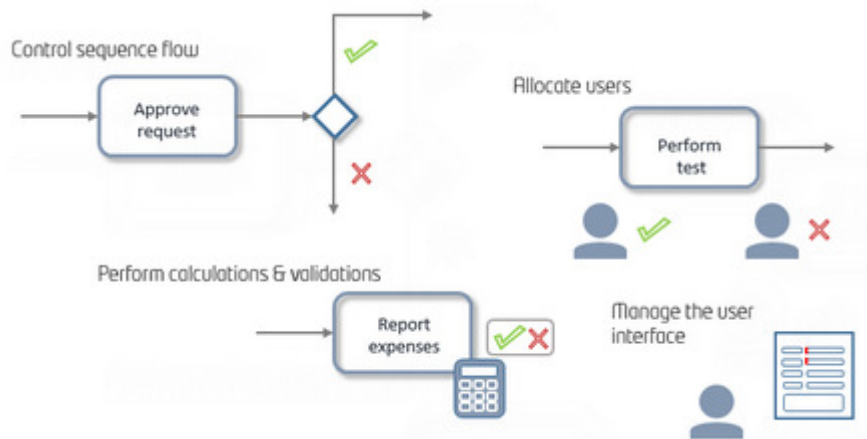


Figure 51: Use of Business rules in Bizagi

### 7.3.5.1 Data access through XPath

All business rules use XPath navigation to help the user traverse the data model in an intuitive way. The XPath standard is an easy-to-use language for finding information in an XML document and is helpful in carrying out complex tasks. Bizagi incorporated this concept into its language, enabling users to easily navigate and use the data model to accomplish different tasks. [1]

XPath as a navigation tool enables users to navigate the data model accessing attributes and relationships. The 'location path' of the desired attribute in the data model is always written between diamond brackets: < xpath route >. Is possible to access any entity, and its attributes, as long as the hierarchy that is laid out in the Data Model is followed. XPath



expressions will always begin with the Process Entity as the first attribute and from there navigate the rest of the data model. Data model navigation is defined by the relationships which are represented by arrows in the data model [1, 5].

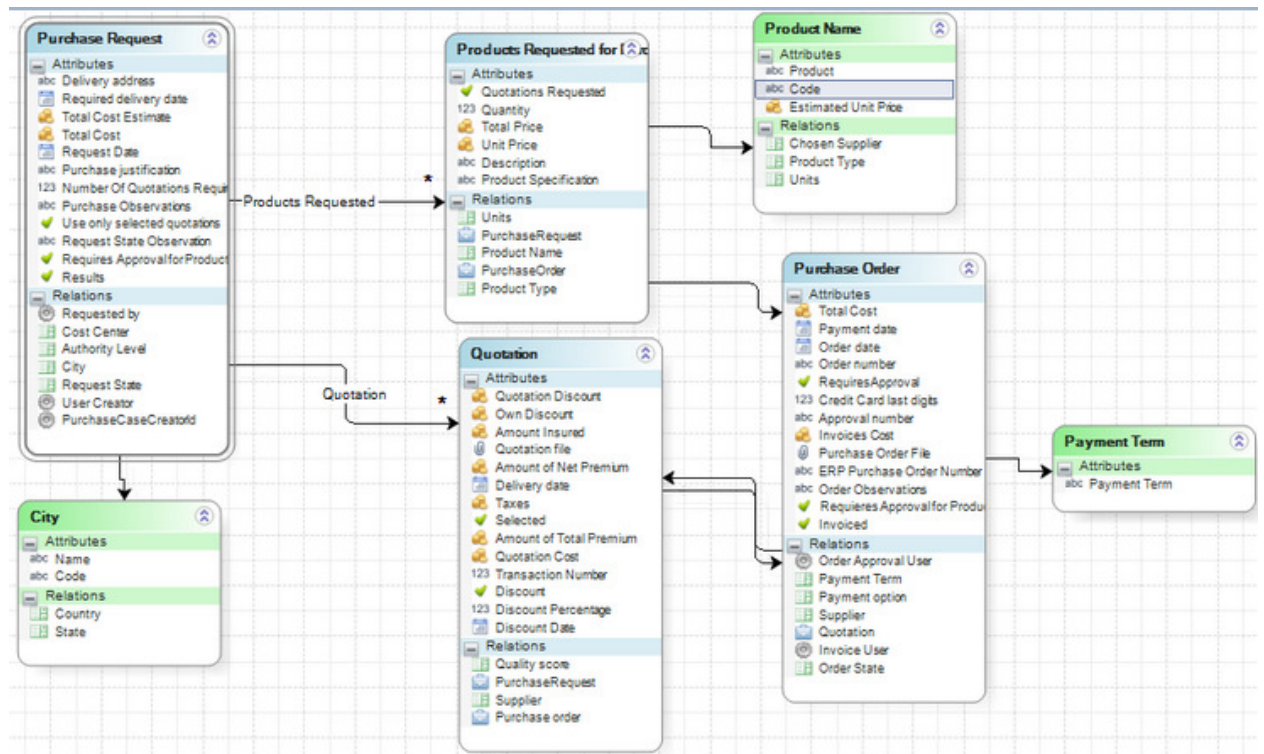


Figure 52: Data model to be used to explain how XPath works

The data model diagram in Figure 52 will be used as a guide to show how XPath works. A Purchase Request process has a process entity called Purchase Request. Consequently, the context is the Purchase Request entity through which the rest of the data model can be reached and is the starting point of the XPath navigation. Using XPath is possible to obtain the value stored in an attribute or set a value to one. To obtain the delivery address of the Purchase Request, the expression: `<PurchaseRequest.DeliveryAddress>` can be used. To get

the delivery date of the Purchase Request, we would use the expression: *<PurchaseRequest.RequiredDeliveryDate>*. To identify the city for this delivery and retrieve its name, is possible to use the expression: *<PurchaseRequest.City.Name>* [1].

It is also possible to navigate collections by means of using the relationship's name. The relationship's name is shown in the arrow connecting the two entities in the diagram and is also demarcated by an asterisk (the non-asterisked paths between entities are the related attribute relationships). To obtain the collection of all Products Requested for the Purchase Request, we could use the expression: *<PurchaseRequest.ProductsRequested>*. To retrieve every Product Name of all Products Requested for the Purchase Request, the expression: *<PurchaseRequest.ProductsRequested.ProductName.Product>* would work for the case. The easiest way to assign a value to an element is using the "=" operator.

*<PurchaseRequest.DeliveryAddress> = "101 Bizagi Road"*; this expression assigns the address to the delivery address attribute [1].

#### **7.3.5.2 Business rules elements**

Scripting rules have different elements that provide structure and facilitate writing code using simple wizards. A rule might be composed of several elements. To include an element, right-click the vertical line of the rule (black arrow) as in Figure 53. This will display a drop-down menu of all elements. Select the desired element and give it a name in the Properties pane on the left. Double-click the element created and select the Properties

option to establish its evaluation characteristics in the new window. When working with the elements, the code associated with the logic expressed in the design is created automatically in the Code View [1].

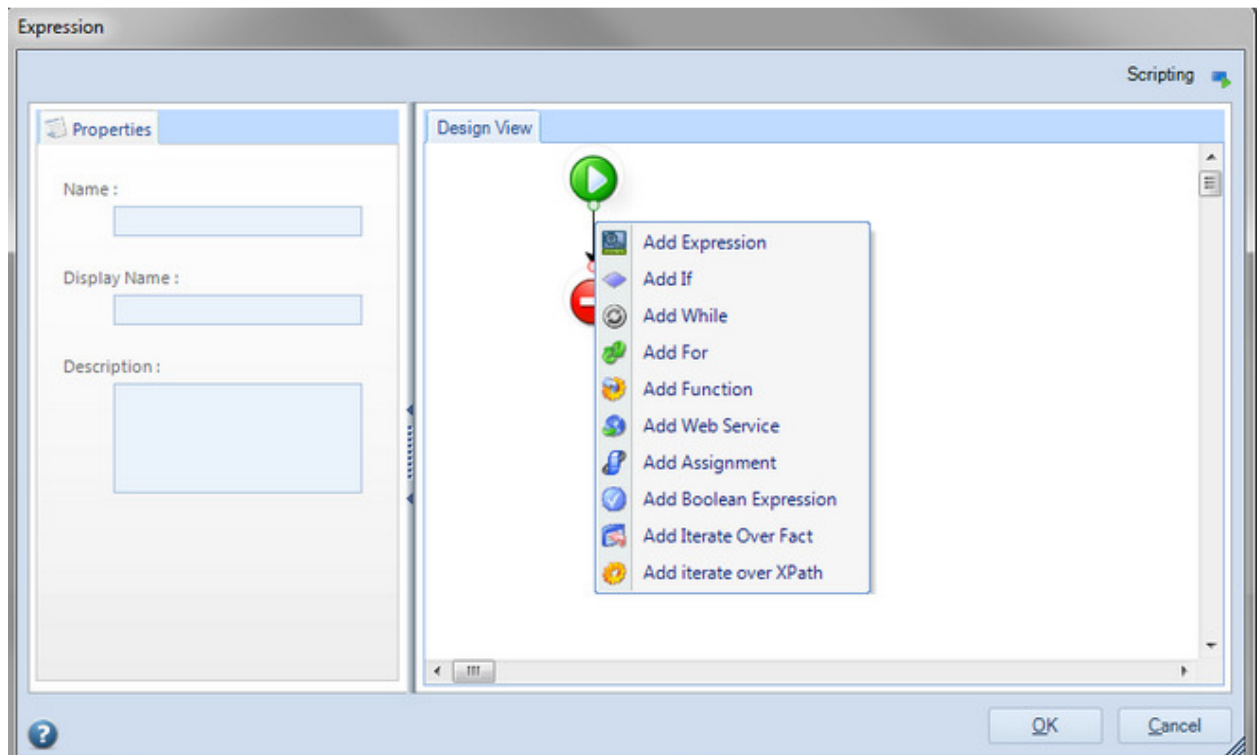


Figure 53: Defining business rules

- *Expression*: This element offers the possibility to type any desired code.
- *If*: Offers the possibility to select one of several paths by means of conditionals.  
The else path will always be established.
- *While*: Executes a cycle as long as a condition is met.
- *For*: Executes a cycle a finite number of times, starting with an initial value and increasing the value by a specified amount on each iteration.
- *Function*: Allows the use of functions created as library rules in each project.

- *Web service*: Include web service calls.
- *Assignment*: Allows value assignments to attributes in the data model using a very simple interface.
- *Boolean expression*: Handles Boolean conditions. When the conditions are met the expression returns True, otherwise they return False.
- *Iterate over fact*: Used to carry out iterations (or cycles) over a one-to-many relationship EXCLUSIVELY when the relationship cannot be accessed directly through the data model using XPath.
- *Iterate over XPath*: Used to carry out iterations (or cycles) over a collection XPath, that is, a one-to-many relationship.

### **7.3.5.3 Business rules examples**

In section 5.3.5 are described the possible uses of the business rules. Let's focus on an example on how is possible to use business rules for process routing. When modeling the process, it is necessary to include the Business Rules that will help determining the path that must be followed by the process flow when reaching the two divergent Gateways that require an expression: Exclusive and Inclusive Gateways. When these shapes have more outgoing sequence flows than incoming ones, they are considered to be divergent. Each outgoing sequence flow must have a business rule associated for Bizagi to evaluate which

path to follow [1].



### **Exclusive Gateway:**

With Exclusive Gateways only ONE business rule can be valid (true). Only ONE path can be followed.



### **Inclusive Gateway:**

With Inclusive Gateways one or more paths can be valid, and those will be followed.

The business rules associated to sequence flows are known as Boolean expressions, since they should always return True or False. On the fourth step of the Process Wizard select Define Expressions [1].



Figure 54: Fourth step of the process wizard, defining business rules

The Process flow will be displayed highlighting the sequence flows that require a business rule to route the Process. When a business rule has been configured, the sequence flow will be dark. When a business rule has not been defined, the sequence flow will be highlighted in yellow [1].

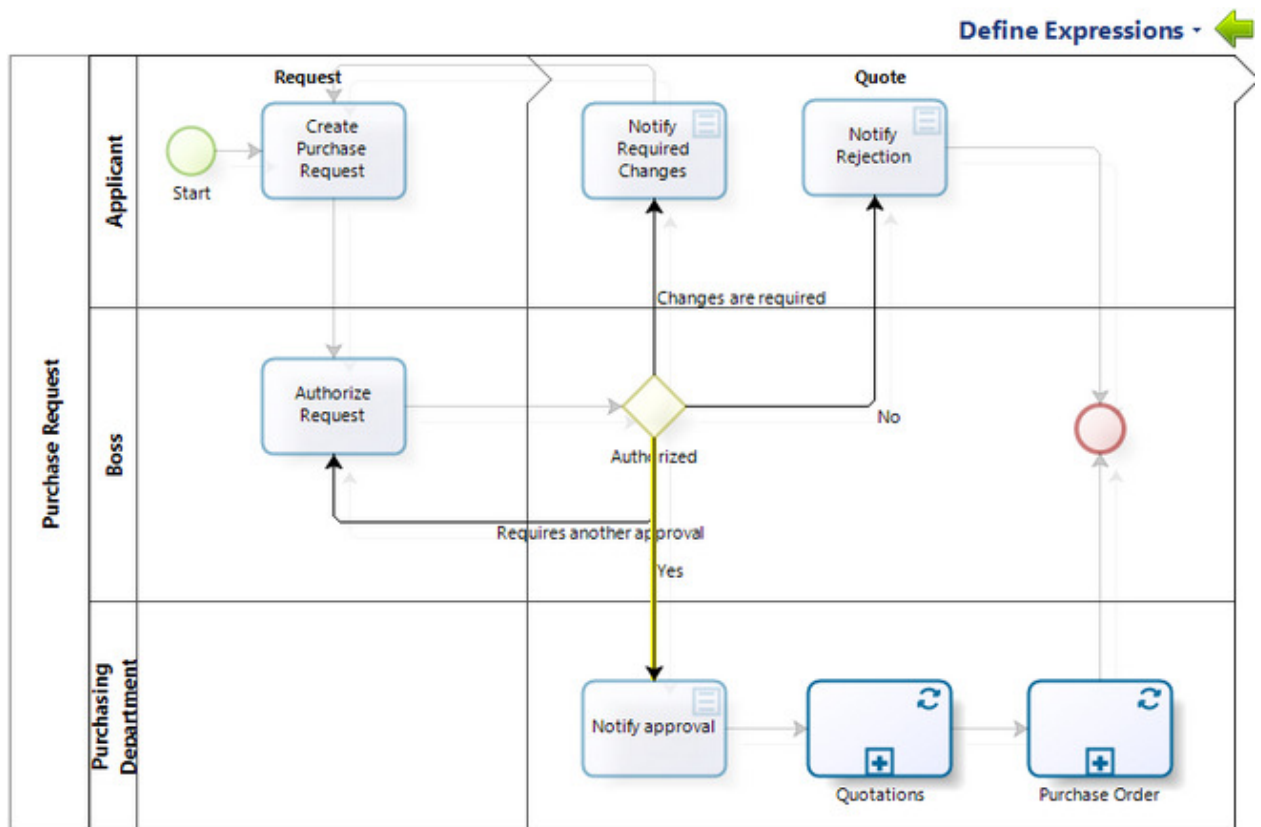


Figure 55: Example process to define business rules in process routing

In a Purchase Request process a user creates a request. Then this request is evaluated by the immediate supervisor (Boss). According to the supervisor's decision the Process flow must follow different paths [1]:

- If it is approved an email will be sent and the process will continue to the Quotations Sub-Process.
- If it is rejected an email is sent and the process is finished.
- If it requires changes an email is sent and the request application is returned to the applicant.

Note that the flow in the Process can only take one path, so we use the exclusive gateway.

For the supervisor to make the decision, he must choose between three options displayed in the drop-down list in the user interface a drop-down list. For this reason, there is a relationship between the Process entity, Purchase Request, and a parameter entity, Request State [1].

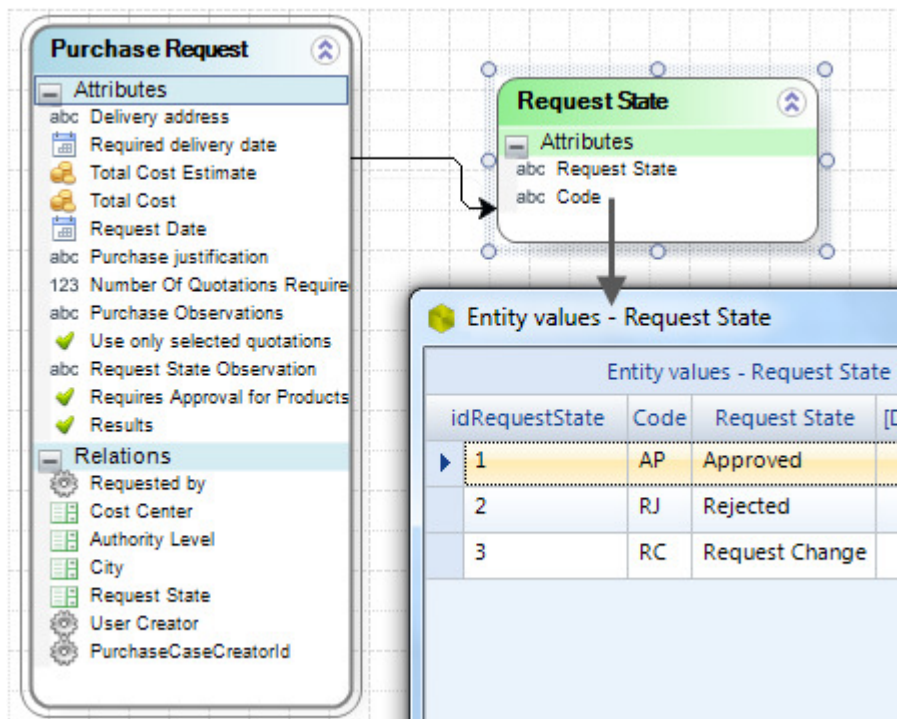


Figure 56: Portion of the data model for the Purchase Request process

Click the sequence flow of the rejected path. In the new window select *Based on the result of an expression*. The *expression tab* will be enabled [1].

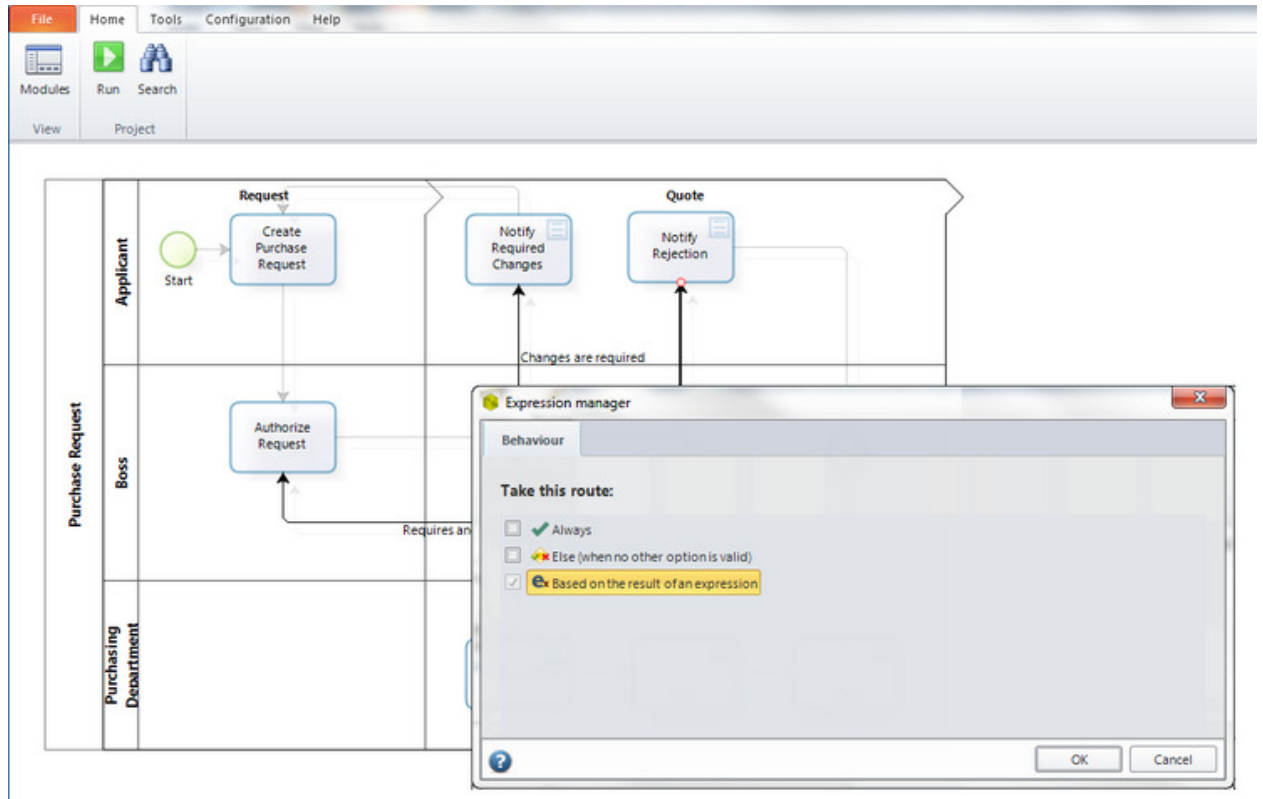


Figure 57: Expression tab

In expression, click *New*.



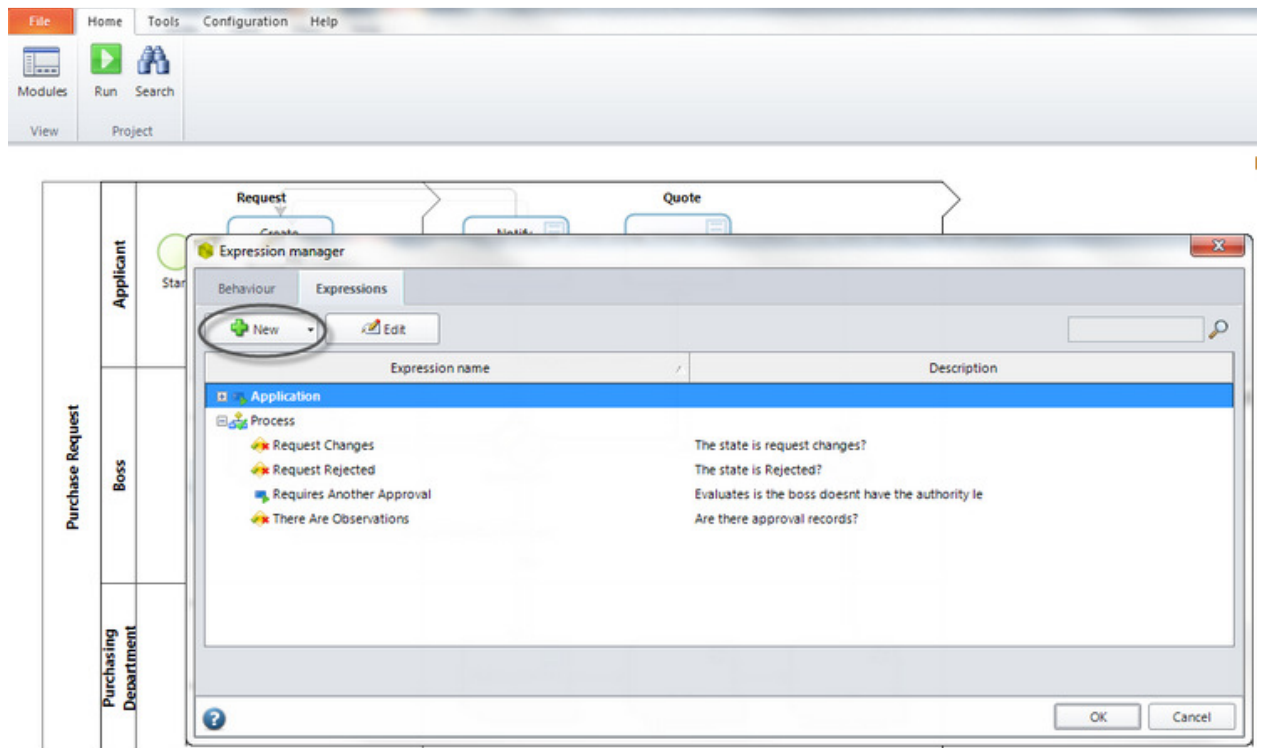


Figure 58: Create a new expression

The *New* tab will display to enter the *Boolean expression*. On the left of the window is the Data Model. On the right side is the space to create the condition that will be evaluated.

- Drag and drop from the data model the attribute to be evaluated. The XPath will be automatically created, in this case: *PurchaseRequest.RequestState*.
- Then, choose the evaluation condition from the drop-down list. When using conditions that evaluate against a parameter entity, as in this case, the list of possible values will be automatically displayed for the user to select from the list.
- Finally select the last element of the condition to evaluate.

Click OK to save the business rule. The steps are also shown in Figure 59 below. Following the same procedure is possible to define the rest of the business rules.

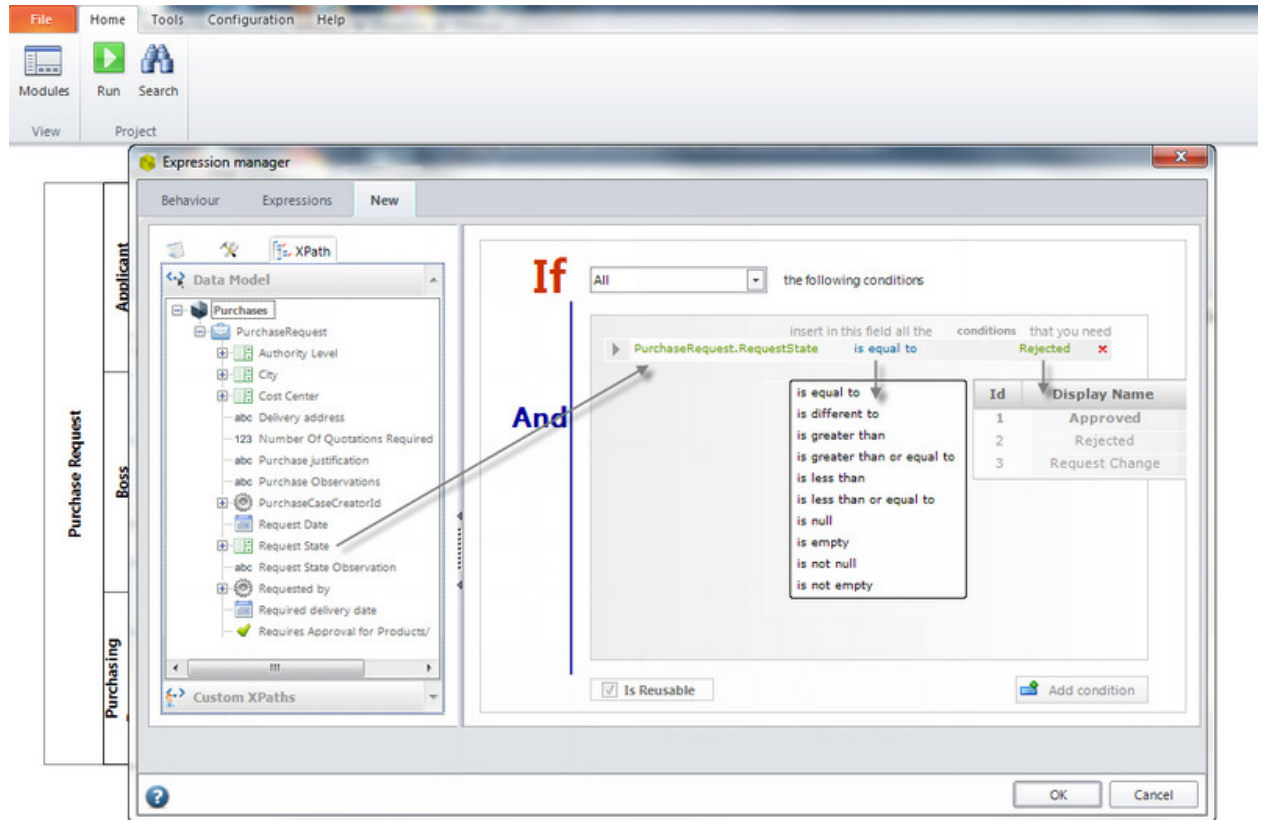


Figure 59: Create a new expression, continued

### 7.3.6 Work allocation

Work allocation is the fifth step of the process automation Wizard where Performers are defined for each Activity of the process. Performers are the users that have the qualities to be assigned to activities. Each Task/Activity created for end user interaction requires definition that will allow Bizagi to allocate the correct users within the organization. Bizagi automatically evaluates the allocation rules defined for each Task and

selects one or more users that meet the given conditions from the user's list. Only these users will have access to work on the Activity allocated to them [1, 6].

To allocate performers, it is necessary to have a user account created for everyone that intends to work with Bizagi. It is important that each user account must be set up correctly to ensure Bizagi selects appropriately. Take into account that if no allocation rule has been defined for a task it will be executed by the user who performed the last task. This powerful feature can be configured based on different criteria like positions, geographical location, skills, roles, among others. Bizagi balances workload among available human resources to further increase productivity. It is also possible to define other distribution methods to suit the particular needs [1, 6].

In the Process Wizard, when clicking on the Performers option, the Process flow will be opened. The different Activities that are available for allocation will be highlighted. The shapes where the performers have not been defined have an exclamation mark. Only Activities and Events that interact with end users will be available to be allocated. This means that automatic Activities like Script Tasks, Gateways or End Events cannot be selected to define allocation rules. Allocations must be defined separately for each Activity and Event [1, 6].

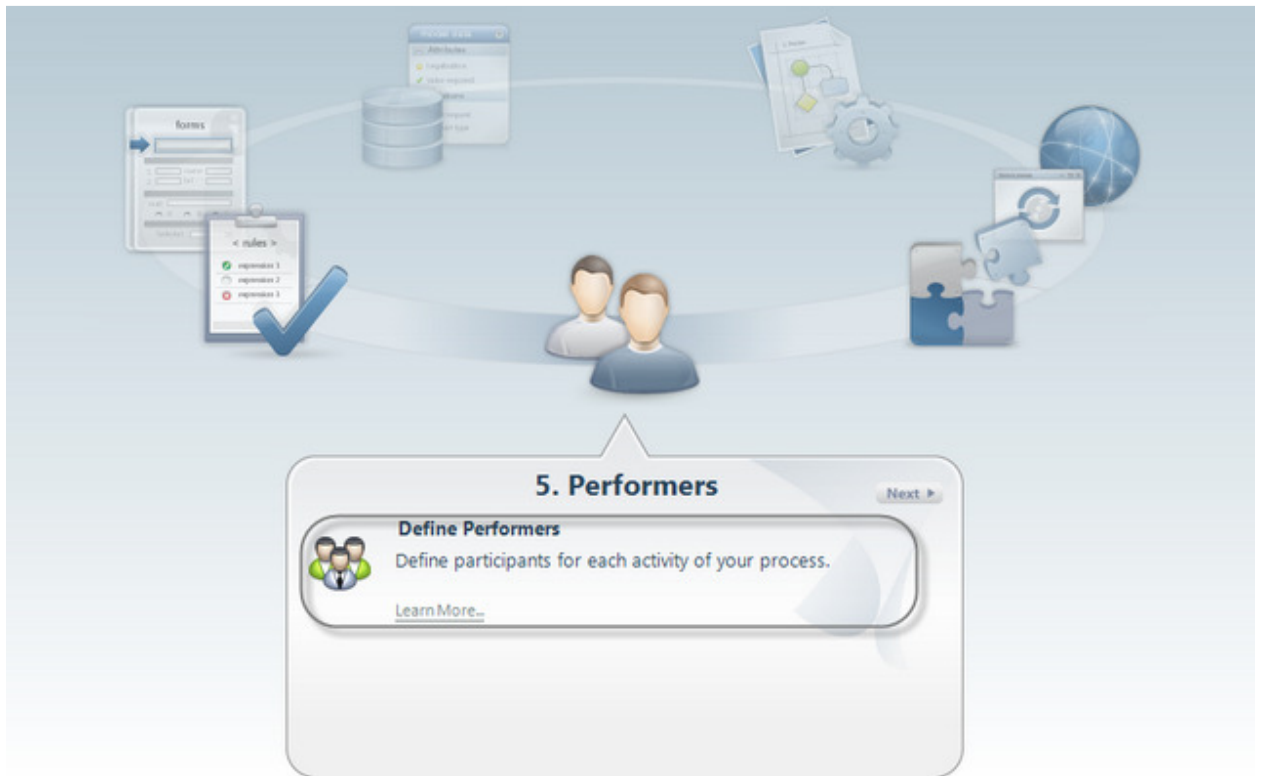


Figure 60: Fifth step of the process wizard, define performers

#### 7.3.6.1 Work allocation components

Once the activity or event is selected, the performers Assignment window will be displayed. In this window is possible to assign an allocation according to three types of conditions: *Allocation Rules*, *Assignment Method* and *Preconditions* [1].

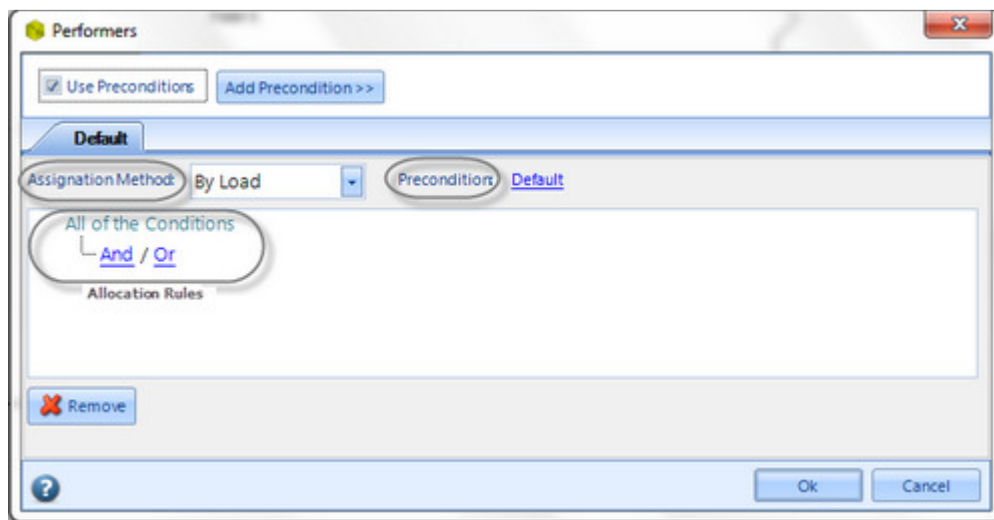


Figure 61: The performer's assignment window

Let see how is possible to define performers using *Allocation Rules*. Determine the specific user or users that should carry out the Activity. To define allocation rules click on *Add Condition* [1, 6].

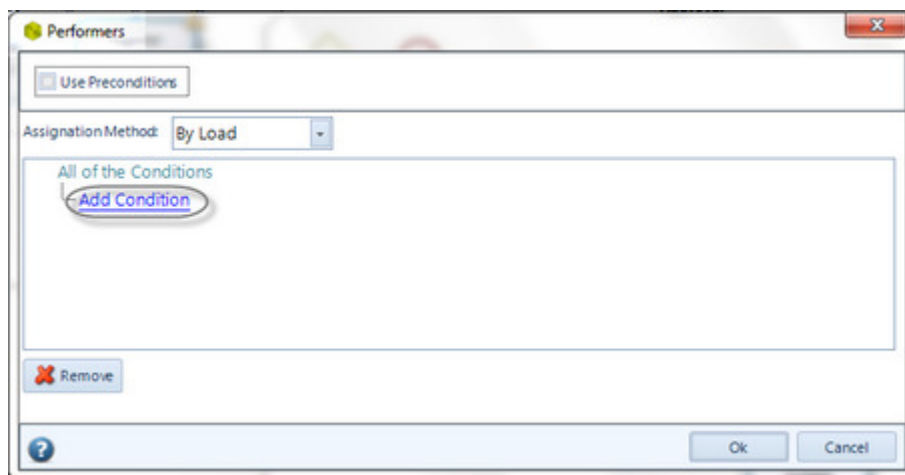


Figure 62: Define performers using Allocation Rules.

Allocation rules are built by specifying operands and operators as shown in Figure [63]:

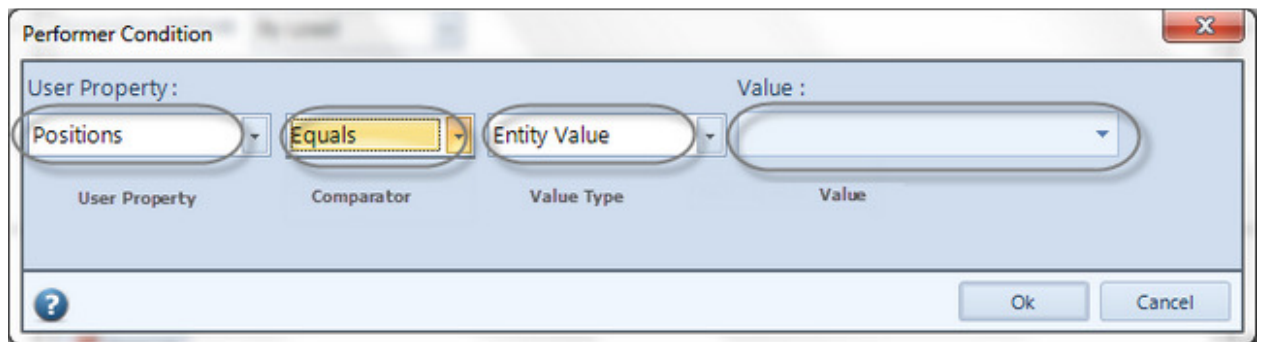


Figure 63: Specifying operands and operators

### User Property

User properties are a set of user characteristics such as name, e-mail, boss, etc. Some user properties are defined by default, but it is possible to create additional user properties according to the specific business needs. The user properties defined by default are [1, 6]:

- *User id*: Identification number in data base of a specific user.
- *Area*: Department of an organization
- *Location*: Geographic location. This section also defines offices or branches, if applicable. In Bizagi, a user can only belong to one location.
- *Role*: Conduct or role carried out by a person in the organization. In Bizagi, a user can have one or more roles.
- *Skills*: Ability or aptitude for an action. A person's special skills allow him/her to carry out an Activity. In Bizagi, a user can have one or more skills.
- *Positions*: Organizational structure. It indicates the positions and their chain of command.

## Comparison Operator

Comparator allows comparing a user property with a specified value. Is possible to select between *Equals to* or *Not Equals to* a condition [1].

## Value Type

Specifies the type of the source where the condition value will be taken from. A type, of the source's value, can be one of the following [1].

- **An Expression:** Allows defining a business rule to establish the value of a condition. Is possible to define an expression or select an existing one from the set of default expressions defined by Bizagi. This source is only available for some use properties such as User id and identification number. Default expressions included in the system are [1]:
  - *Current Assignee*: Returns the user currently working on the Process.
  - *Current Assignee Boss*: Returns the boss of the user currently working on the Process. This expression has variations as *CurrentAssigneeBoss\_Level2*, *CurrentAssigneeBoss\_Level3* etc to access different hierarchical levels.
  - *Case Creator*: Returns the user who created the case.
- **A Data Binding:** Allows selecting an attribute in the Process Data Model.
- **An Entity Value:** Allows selecting the value from a list of values displayed for the Parameter Entity.

## Value

Is the value of the allocation condition which is set according to the value type selected.

One or more conditions by using the And/Or operators can be specified [1].

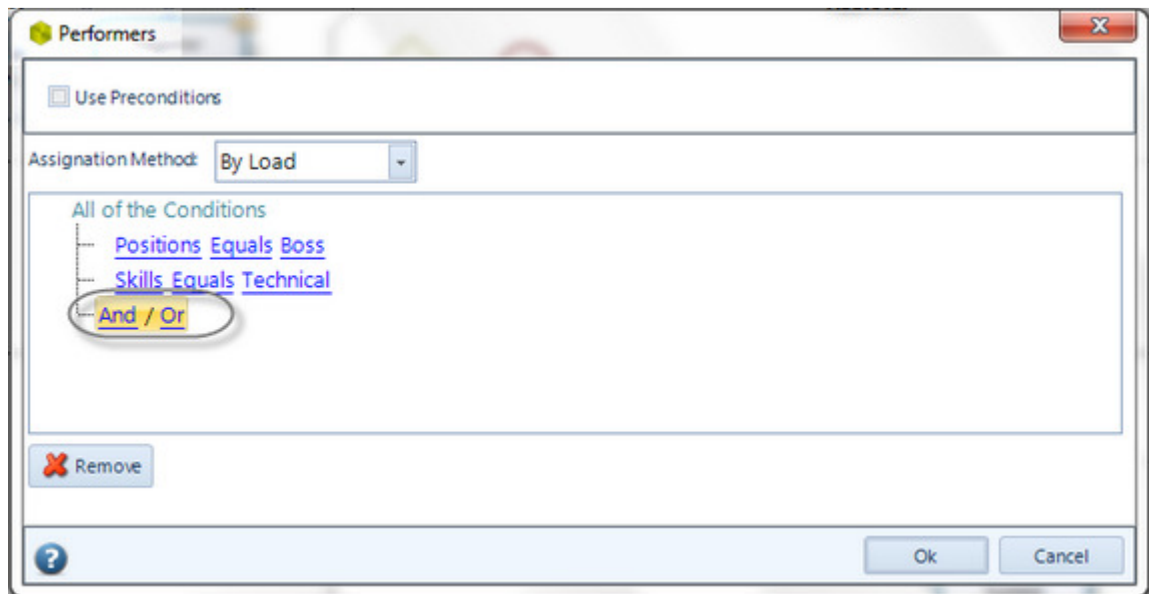


Figure 64: Define performers using Allocation Rules example

### Assignment Method

Provides a set of functions to select how Bizagi allocates a Task to the available users. The four options available are [1]:

- *By Load*: The Task is allocated to the user with the lightest workload or with the least “jobs pending” on the project. However, the system first checks whether anyone in the user's group has already worked on the case. If so, he/she is assigned the Task, regardless of the user's workload compared to the rest of the group.
- *Everyone*: Allocations are given to all users with the characteristics indicated. The first person to take the case (i.e., click on it in the pending inbox) will carry out the Task; consequently, it will no longer be displayed for the others to see.
- *Sequential*: Each Task is assigned evenhandedly and sequentially among the users who satisfy the allocation criteria, regardless of their workload. However, if a Task



is instantiated several times, this Task will be assigned to the same user that was allocated the first instance, and the assignation method will not apply in this case.

- *First available user:* With this method, tasks are allocated to the user that will first be available according to the time zone associated. There may more than one user available. In such cases, Bizagi will determine the task performer by evaluating the workload of each available user [1].

### **Precondition**

Allows carrying out the allocation based on business rules. These results return a result of true or false, indicating whether or not the condition that applies to the defined profile was met. In other words, a precondition allows a user to establish rules in order to decide which allocation rule to follow. Preconditions are activated by checking the Use Precondition box. Each tab represents a precondition and contains its own allocation rule and assignation method, this way an Activity allocation can be done by a combination of assignment rules and assignation methods according to the business conditions [1].

#### **7.3.6.2 Basic task allocation**

Let's describe how to set basic allocation rules (without using preconditions). The process in consideration is the *Vacation Leave Request*. In this process an employee enters a vacation request. It has to be approved by the boss and, once approved, must be recorded in the company's payroll system. The respective process model is depicted in Figure 65 [1]:

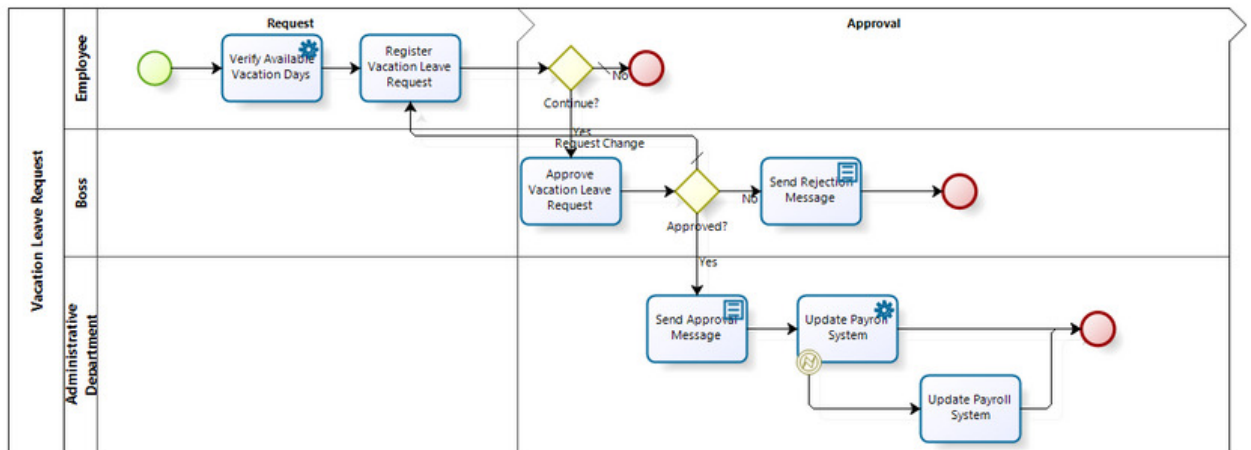


Figure 65: Vacation Leave Request process model

Let us consider the following configurations [1]:

- Assign the *Register Vacation Leave Request* activity to the person who created the case.
- Assign the *Approve Vacation Leave Request* activity to the Boss of the person who created the case.
- Assign the *Update Payroll System* activity to an Administrative Assistant

1. Go to the fifth step of the Wizard (Performers) and click on Define Performers
2. Select the desired Activity to define its Performers.

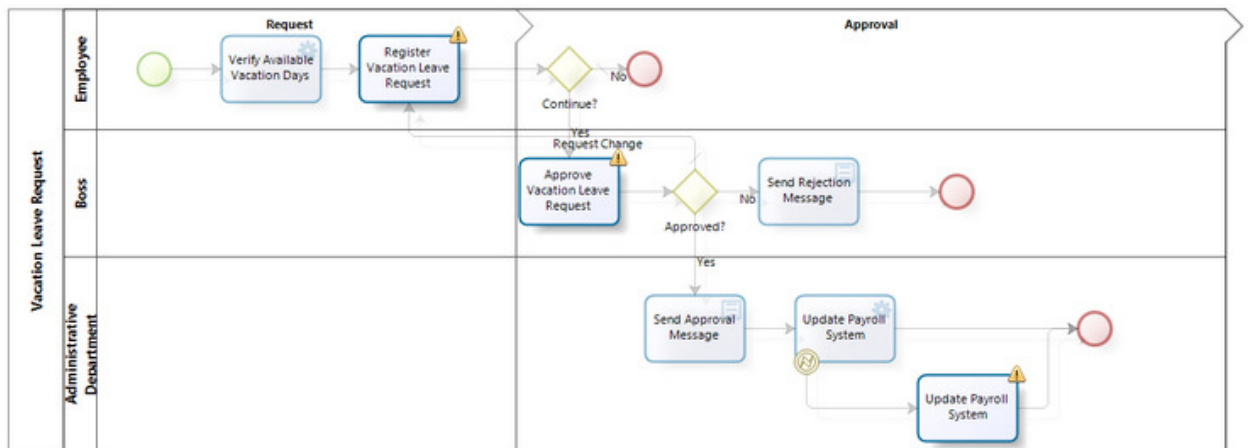


Figure 66: Vacation Leave Request process model, in define performers window

- Once the Activity is selected, the Performers allocation window will be displayed.
- Click on *Add Condition* to include a condition where we select the properties that a user must meet in order to be allocated.

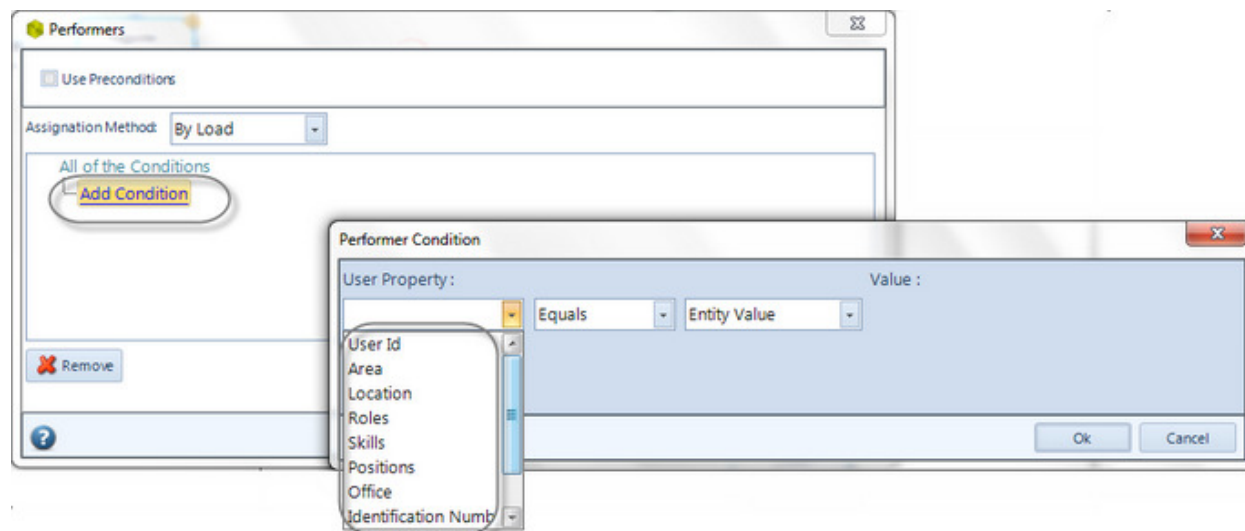


Figure 67: Define performers using allocation rules

For the *Register Vacation Leave Request* activity we are going to allocate it to the case creator. Select *User Id* from the *User* property list The *User* Property is compared with an expression that establishes the case creator ID. Select *Equals* in the Comparator field and *Expression* as the source type in the Value Type field. Click *Select Expression*. The Expression editor will be opened, here you can establish your own business rules. Let's use one of the predefined business rules. Click on *Cancel* in the Expression editor. A new window will be displayed to choose the pre-defined and reusable business rules. Select the *Case Creator* expression and click on OK [1].

4. For the *Approve Vacation Leave Request*, repeat the same procedure stipulated above to allocate the Register Vacation Leave Request activity, but in this case select the expression *CurrentAssigneeBoss* [1].

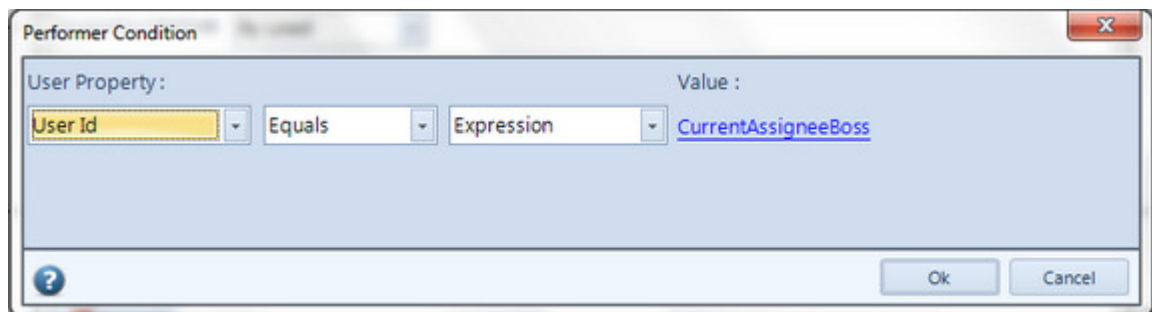


Figure 68: Define performers for the Approve Vacation Leave Request activity

5. For the Update Payroll System Activity we will allocate this task based on the position of the performer. Select *Positions* from the *User Property* list.

Select *Equals* in the comparator field and *Entity Value* in the Value source field.

Here the entity value is a position within the organization. Select the *Administrative Assistant* value from the *Position* values in the *Organization* entity [1].

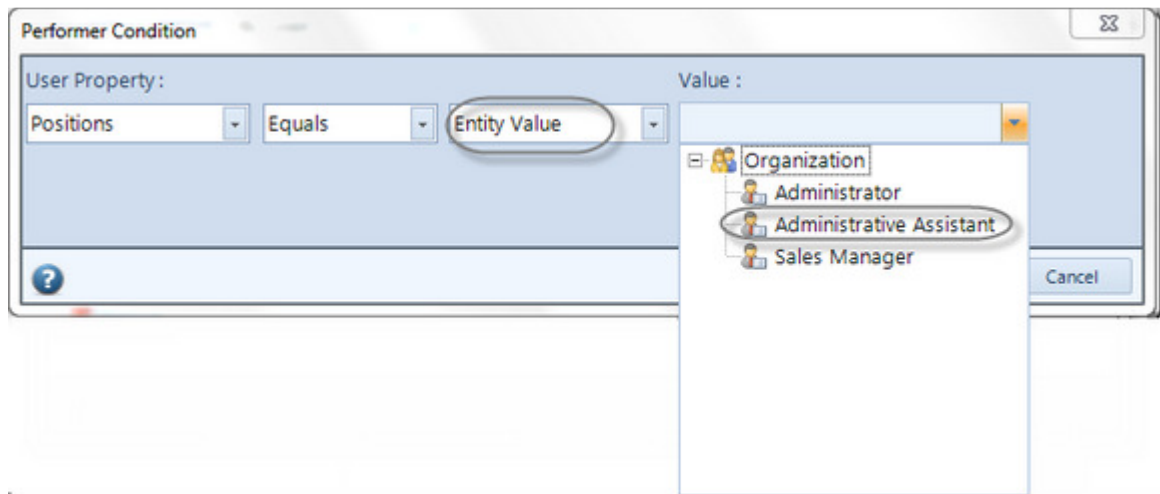


Figure 69: Define performers for the Update Payroll System activity

5. In the event that the *Entity Values* have not been defined, is possible to allocate a new value by clicking on the *New* link. However it is strongly recommend to define the organizational structure first. In the next section will be provided more information on how to administrate organizations [1].

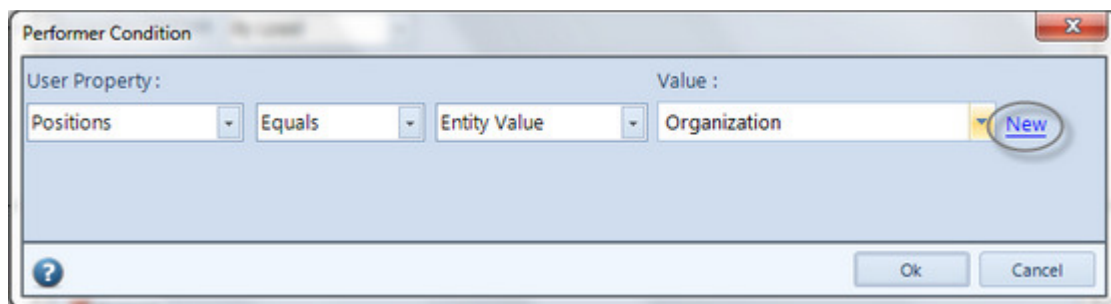


Figure 70: Define performers for the Update Payroll System activity, define the Entity values.

6. **Select the Assignment Method.** For the *Register Vacation Leave Request* and *Approve Vacation Request* activities, the assignment method is not relevant because there will only be one user that meets each allocation rules (*Case creator* and *Case creator Boss*). For the *Update Payroll System* activity select *By load*. This means the activity will be assigned to the Administrative Assistant with lowest work load. Finally click on Ok [1].

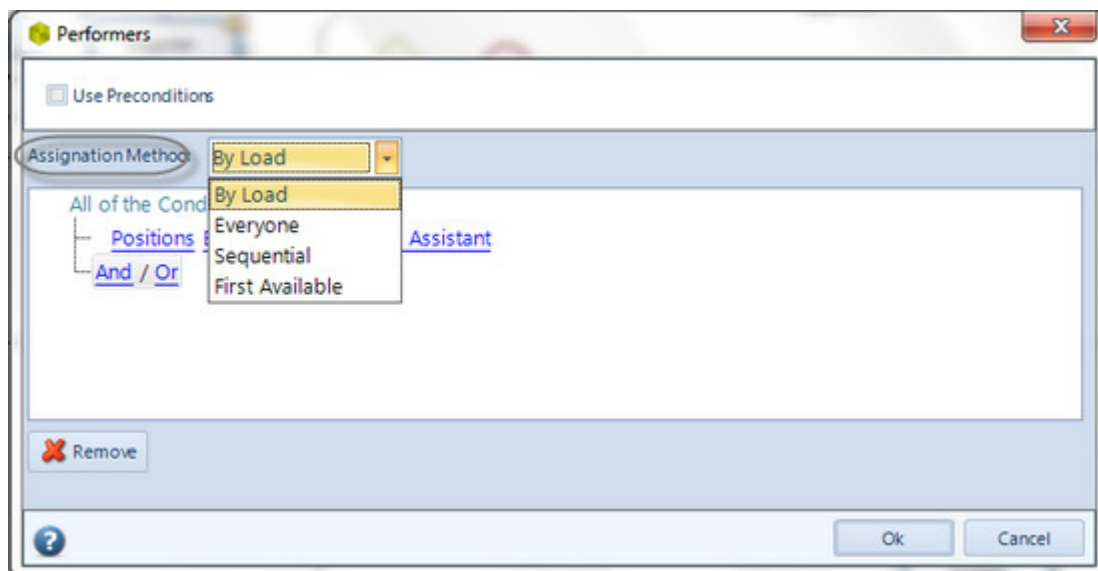


Figure 71: Select the Assignment Method

### 7.3.6.3 Organization

Organization in Bizagi is a feature where is possible to define the hierarchical structure of the company and the association between the different people or areas of which it consists. An organization in Bizagi stores the information related not only to the organizational structure of the members of a company and the definition of their characteristics (*position, areas, groups*), but also the characteristics that make them unique on a team and allow them to be active members in the processes of the application or applications (*roles, skills, geographic location*) [1].

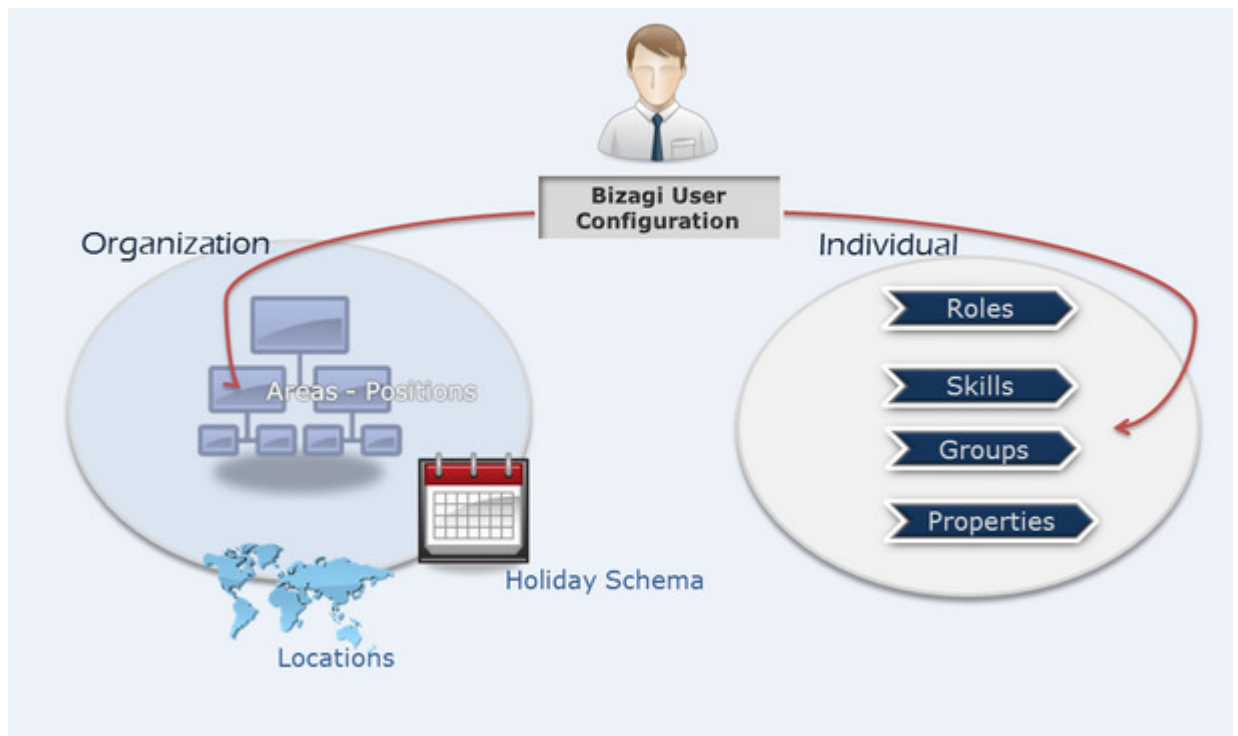


Figure 72: Organization in Bizagi.

The definition of an organizational structure is vital for task allocation and security management. As we discovered in previous sections, allocations are done based on user properties which are defined in the Organization Module. Likewise permissions to create

and administrate a process and its components are granted according those properties. In this section we show how to configure organizations [1].

## **Organization Components**

Organization definition is done through the configuration of several components. These components define characteristics for the global organization and individual users [1].

## **Organizational Structure Components**

These components define the main attributes of the organization, allowing to locate users within them [1].

- *Areas*: Departments of an organization. In Bizagi, a user can only belong to one Area.
- *Position*: It indicates the positions and their chain of command. In Bizagi, a user can have one or more positions.
- *Location*: Geographic location. This section also defines offices or branches, if applicable. In Bizagi, a user belongs to one location.
- *Holiday Schema*: Contains the calendars of working and non-working days that apply for the organization. It is possible to define different schemas to adjust to the fact that each country, region or city has different non-working days. In Bizagi, a geographic location has one Holiday Schema.



- *Working Time Schema*: The Working Time Schema refers to the work schedule (from 8AM to 6PM and so on). The expiration of processes and tasks depend on the working time schema. These are denoted with stop lights on the Work Portal and are used for the calculation of the Analytics reports. In Bizagi, it is possible to define one Working Time Schema for each organization [1].

### **Individual User Components**

These components define the main attributes used to describe particular characteristics of users [1].

- *Roles*: Conduct or role carried out by a person in the organization. In Bizagi, a user can have one or more roles.
- *Skills*: Ability to perform actions. In Bizagi, a user can have one or more skills.
- *User Properties*: A set characteristics such as name, e-mail, immediate superior, etc. Is possible to define additional user properties required for processes or assignments, or simply as additional user information.
- *Time Zone*: The 24 main regions that allow a user to identify the Local Time according to the position of the city or country in the World.
- *User Groups*: User profiles or a set of characteristics of an employee of the organization that allow to define levels of access to applications and features [1].

### **User's Configuration**

When the components of an organization have been defined you must assign each end user their specific characteristics. These configurations are done through the *Work Portal* in the *Organizations* and *Configuration User* tabs of the *Users Menu* [1].

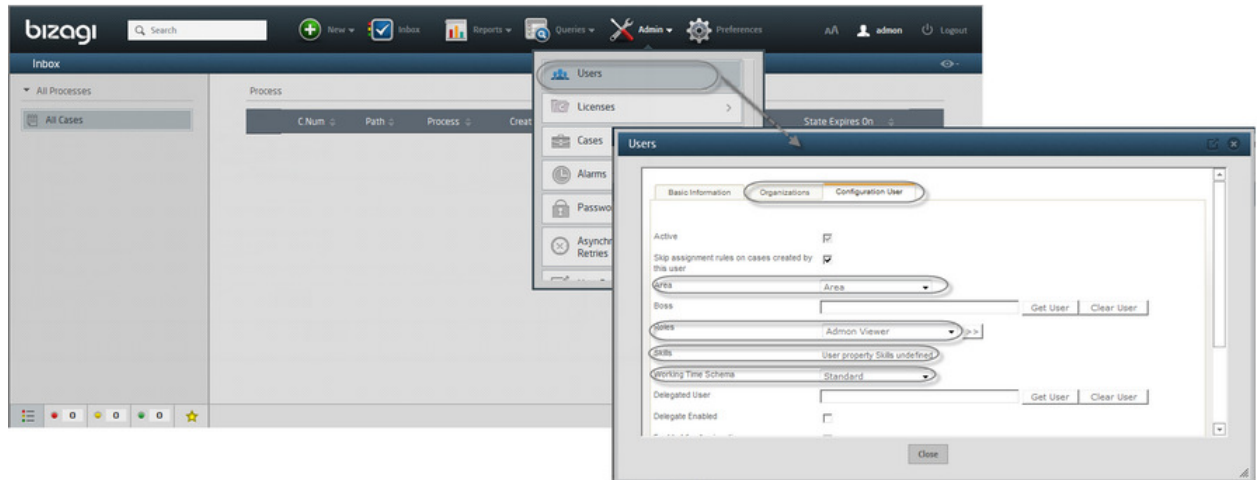


Figure 73: Bizagi Work Portal

### 7.3.7 Application Integration

Application integration is the next step of the process automation wizard. Bizagi provides a powerful integration layer that supports the different integration possibilities involved in a complete BPM enterprise solution. In such enterprise projects, where there is the need to integrate existing services and applications, Bizagi's Integration layer presents the following diverse options. More information and specific details regarding application integration is present on the Bizagi Studio documentation [1, 7].

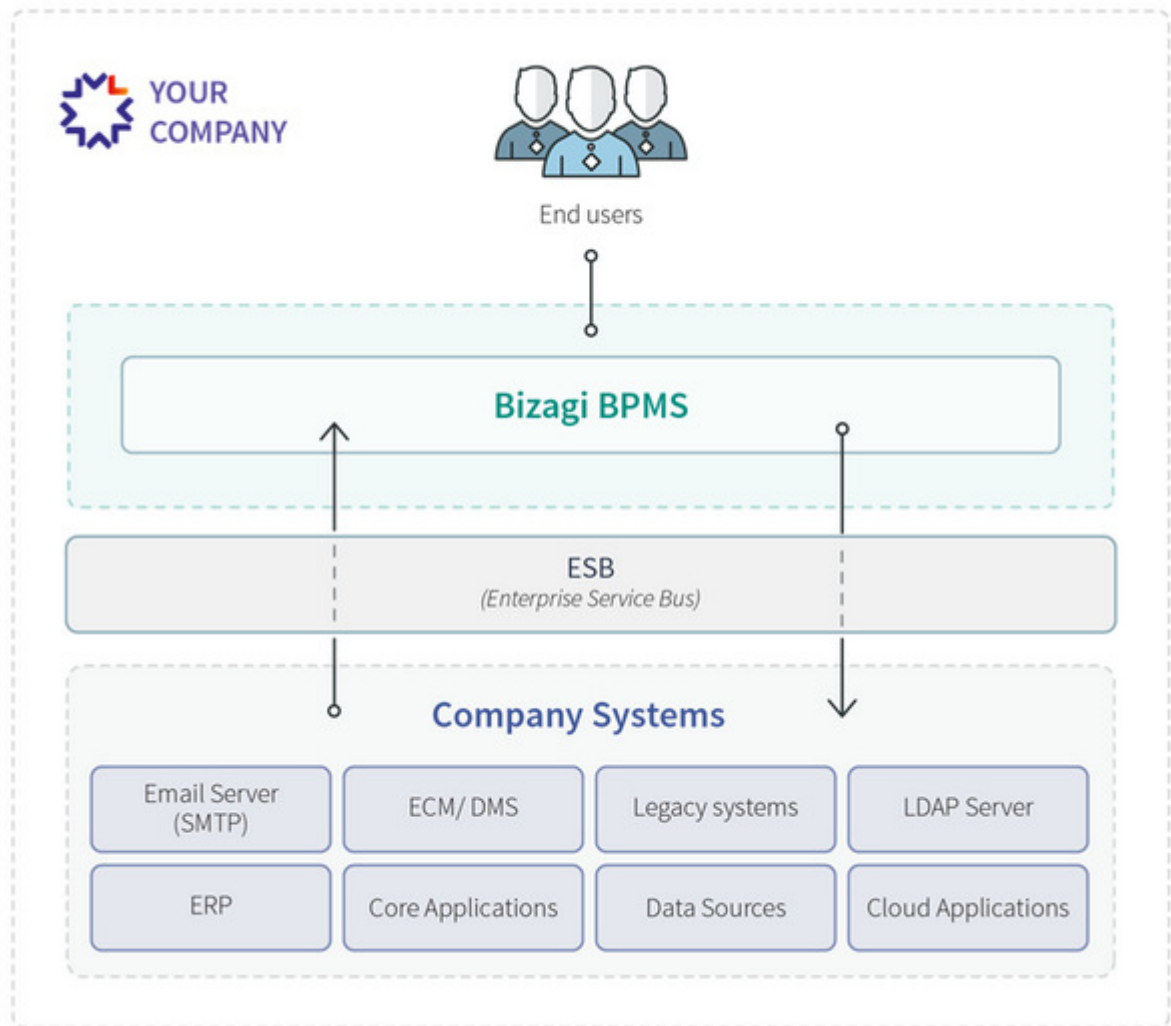


Figure 74: Bizagi integration capabilities

### Invoking external services from Bizagi

For modern and service oriented applications, Bizagi features a Web service connector which is easily configured to consume SOAP Web services or REST services (i.e, WCF services, RESTful APIs, any SOAP service available through the ESB, on-premise, or available at the cloud), in an asynchronous or synchronous fashion. This is all done through

assisted steps, by means of graphical mapping and configuration (without the need of programming) [1].

### **Integrating APIs or custom code in Bizagi**

Is possible also extend the logic behind business rules in Bizagi, by including specific own connectors. You may write you own code to directly integrate APIs, or to customize how you connect to legacy systems to fetch or update information. The concept behind this feature, is to provide a separate component in .NET or in Java technology, and include this component as a class library in Bizagi [1].

### **Additional integration mechanisms for your servers or databases**

Bizagi offers other integration features for the corporate BPM Solution. These features involve options such as the possibility to connect to external data sources, storing documents in an ECM repository, and LDAP integration, amongst others [1].

## **7.4 Bizagi Engine**

Once all the above process automation steps have been completed in Bizagi Studio, the automated process is ready to be executed in the Bizagi Engine. Bizagi Engine executes the automated processes and delivers them to the desktops and mobiles of every business user. The main characteristics are [1, 8]:

- *Built for performance*

Bizagi BPM platform is designed for the most demanding enterprise needs, capable of handling mission-critical, high performance BPM projects spanning thousands of users and millions of cases.

- *Meet organizational performance objectives*

Visualize process objectives, set priorities and spot and correct any issues. Keep on top of what matters– all from within a personalized web portal.

- *Integrate with any IT assets*

Bizagi Engine offers unique support for native JEE or .NET platforms ensuring seamless and effective integration with existing IT investments.

- *“Always On” performance*

Bizagi’s clustering technology means that business processes will continue to operate normally – no matter what happens.

- *Achieve continuous improvement*

Get deep organizational insights with the available reports, right out the box. See who’s doing what, when and where with intuitive graphical tracking and monitoring tools. Bizagi Engine ensures every step in the process flow takes place exactly as it should.

- *Anticipate problems and manage risk*

Historical performance and data trends are available giving the possibility to quickly identify bottlenecks and resource challenges – before they become an issue.

- *Manage by exception*

Bizagi Engine raises the alarm the moment there are exceptions to the rule. Service Level Agreements or non-compliance issues are reported to the right people in an efficient way [1, 8].

Bizagi presents a Work Portal to end users where the process model, previously designed and constructed, is interpreted and executed. Use of the Work Portal requires a browser (intranet or Internet) and a registered account. The Work Portal is where end users access allocated cases or create new cases for a given process, to control the cases processing according to the business logic design. A case is a process instance and so, when an end user starts a new case, he/she will create an instance of a process. All instances are completely independent from one another. The Work Portal resulting from Bizagi's automation has a very important feature: when the process is modified (i.e., any element of the business model) the application adapts itself automatically. That is why, in Bizagi, ***the Process IS the application*** [1, 8].

Depending on the user profile, this portal also enables management of the processes, reassignment of cases when operational problems arise and taking corrective actions for achieving organization's efficiency. The main screen of the Work Portal is divided into three sections; *Main Menu*, *Cases Area* and *Work Area* as shown in the below Figure 75 [1, 8].

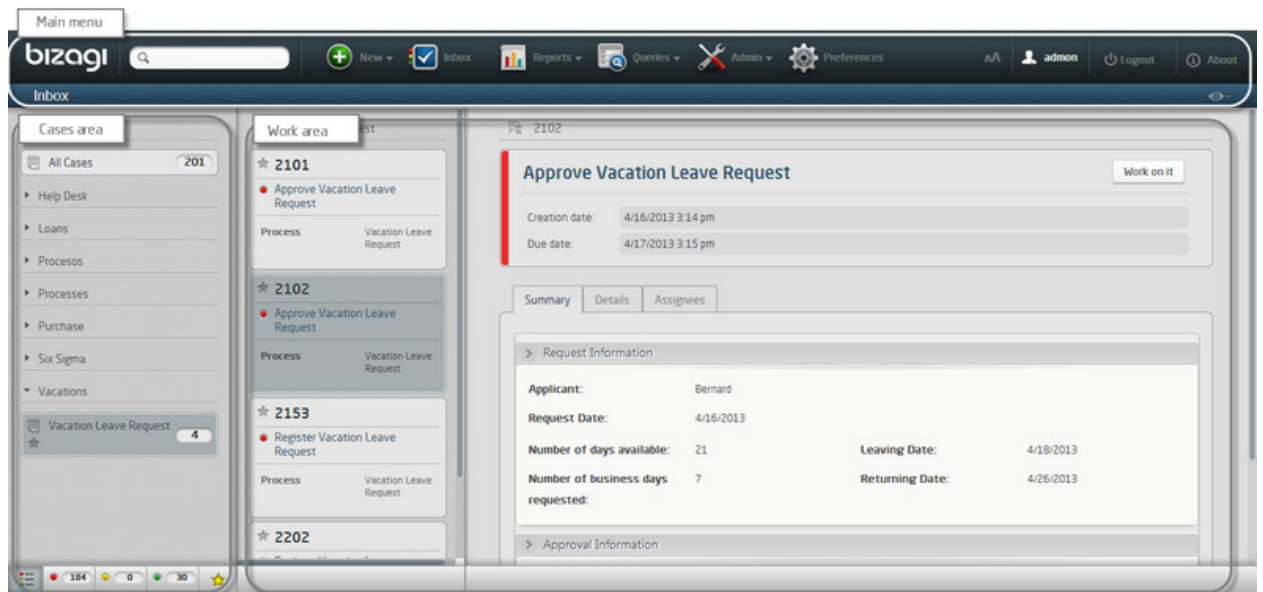


Figure 75: Bizagi Work Portal

To access the Work Portal click the Run button in Bizagi Studio's Home Tab or in the seventh step of the Process Wizard [1].

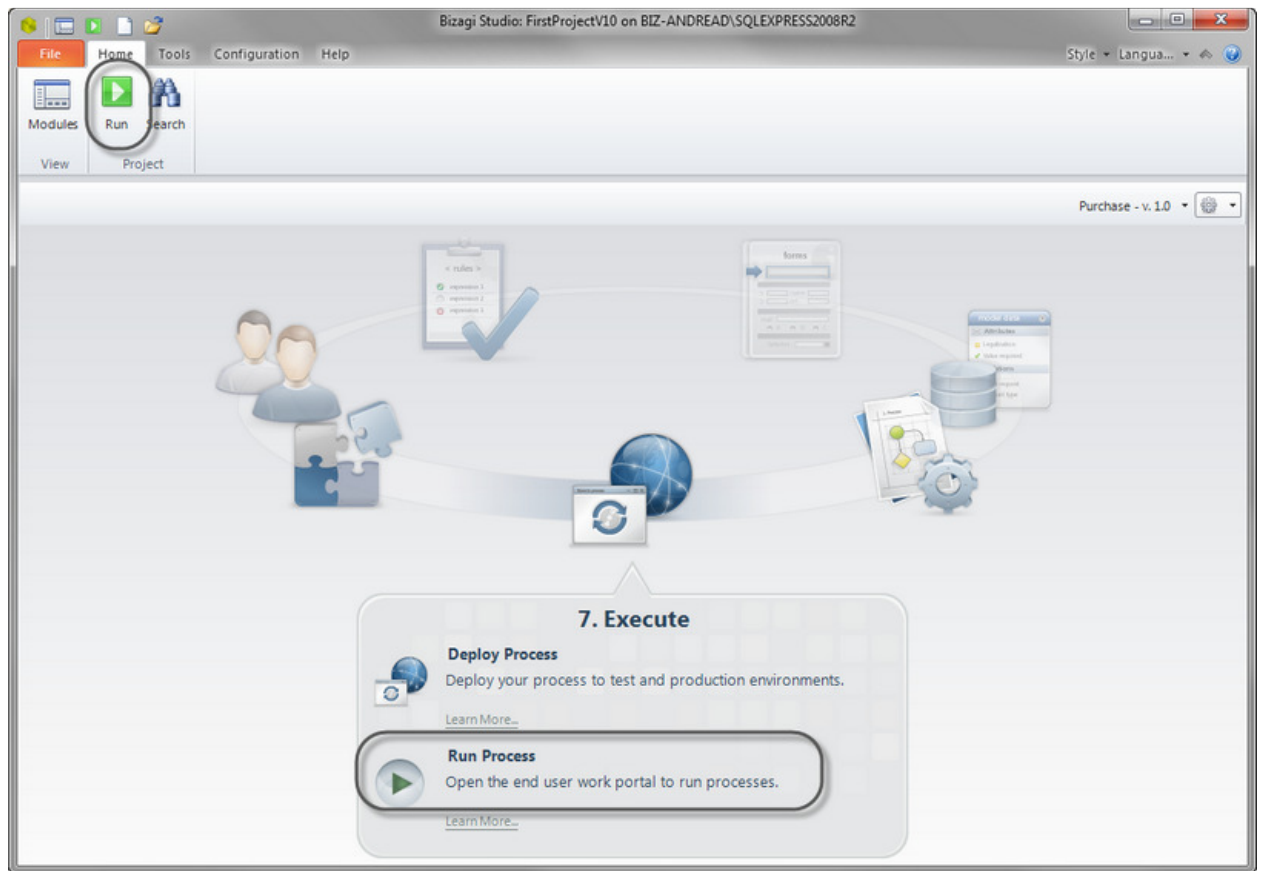




Figure 76: How to access the work portal in Bizagi studio

## CHAPTER VIII: MOSKITT

### 8.1 MOSKitt framework

Modeling Software Kit (MOSKitt) is a free software framework built on the Eclipse platform developed by “*Conselleria de Infraestructuras, Territorio y Medio Ambiente*” in Valencia, Spain [37]. In particular we will focus our attention in “MOSKitt Code Generation” module which supports the semi-automatic generation of OpenXava [46] applications through a chain of model transformations from models (UML models, User Interface Models) to code. The model transformation functionalities provide automatic generation of the OpenXava code necessary to ensure the persistence of the entities in the database and the generation of the AJAX user interface. An enterprise web application with create, read, update, delete and export functionalities is automatically generated and operational. The semi-automatic generation refers to the fact that the OpenXava code generated automatically must be completed by the developers to include specific business logic [31].

OpenXava is an open source AJAX Java Framework used for Rapid Development of Enterprise Web Applications [46]. The Model Driven approach supported from MOSKitt Code Generation will be presented through the development of a specific business information system in the latter sections.

MOSKitt Code Generation module comes with a development environment called OXPortal that deploys and executes OpenXava applications. Basically it's a compressed package that contains the following components: Apache Tomcat container, Liferay Portal,

HSQldb database etc. To setup and make operational the developing environment is simple and intuitive [32, 33].

## 8.2 Set up and installation

Set up the MOSKitt framework is easy and intuitive since it is based on the Eclipse platform [21]. The MOSKitt framework is available for free for both Windows and Linux platforms and can be downloaded from the MOSKitt home page [37, 33]. To install the tool:

1. Unzip the version for the specific operating system. A folder named “moskitt” will be created
2. Copy and paste the unzipped folder in the folder designated to save it

In this way, the tool can be used from any folder or device on which the user has writing and reading permissions: pen drives, USB hard drives, etc. Once the unzipped folder is saved, double-click on the icon called “*MOSKitt*” to start the tool. NOTE: in Linux, it may be necessary to grant execution permission to the “*MOSKitt*” file [33].

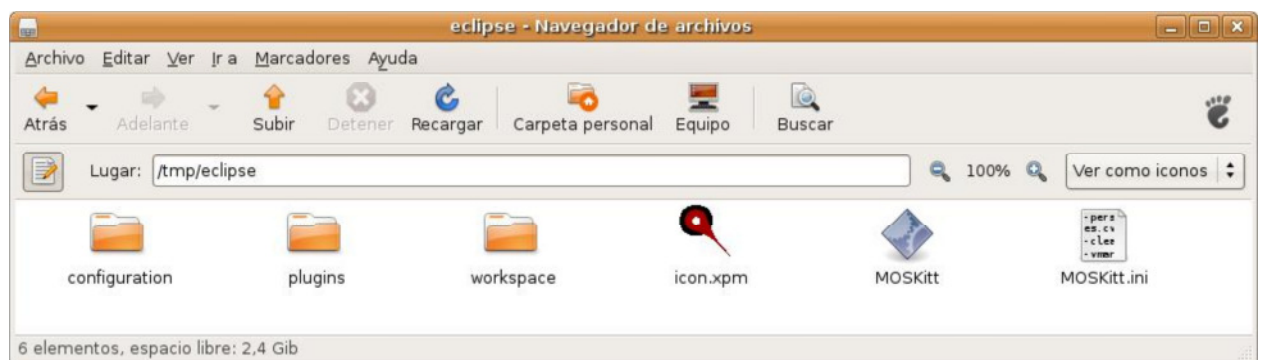


Figure 77: MOSKitt folder

The first step is to select the location of the workspace to be used. In order to create a new workspace, indicate the location of a directory which does not exist. In this case, the tool creates the folder and the workspace. In the case that the user indicates the path of an existing workspace, its content is loaded automatically [33].

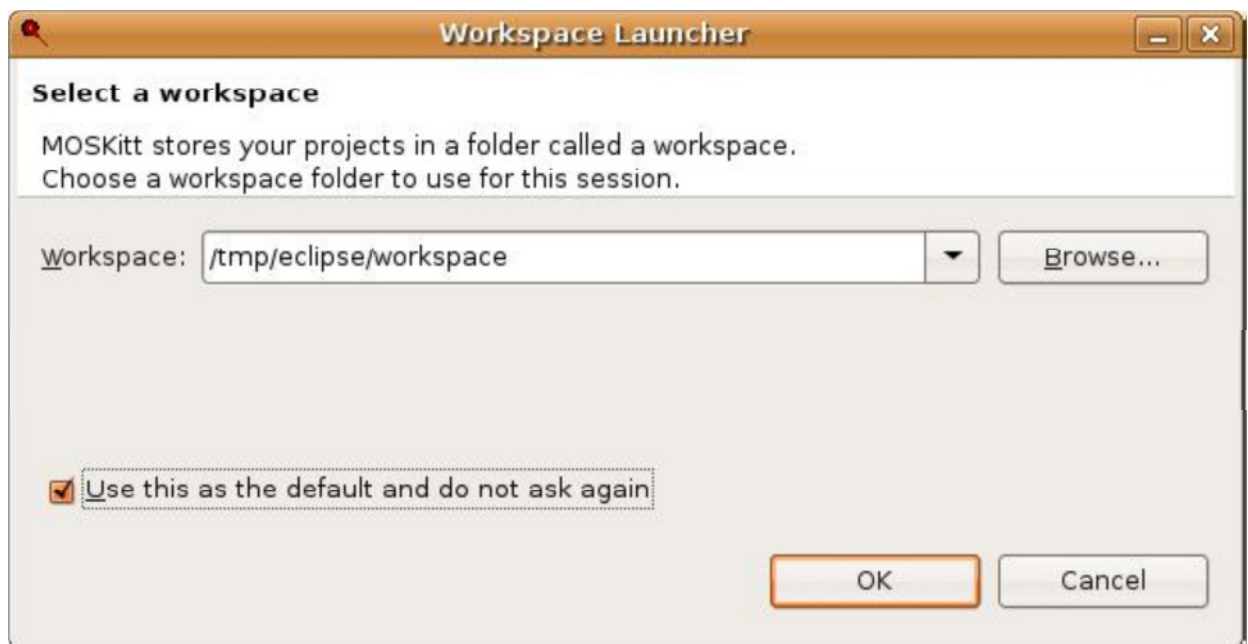


Figure 78: Defining the MOSKitt workspace

When you enter for the first time, a workspace is created and the MOSKitt environment is displayed [33].

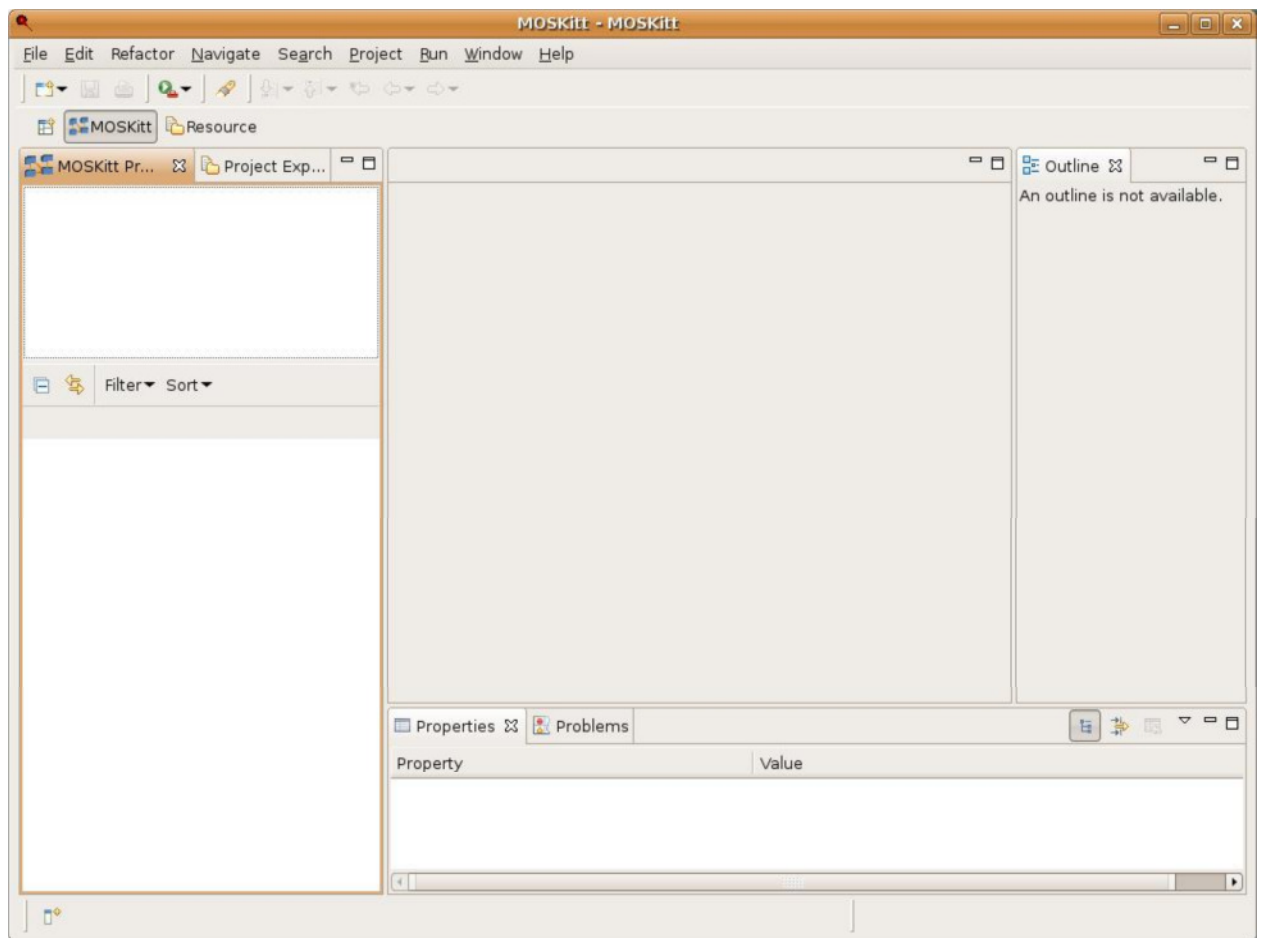


Figure 79: MOSKitt environment in execution

To start working in MOSKitt, it is necessary to create a new project. To create a new project, select the menu option **File -> New -> Project** [33].

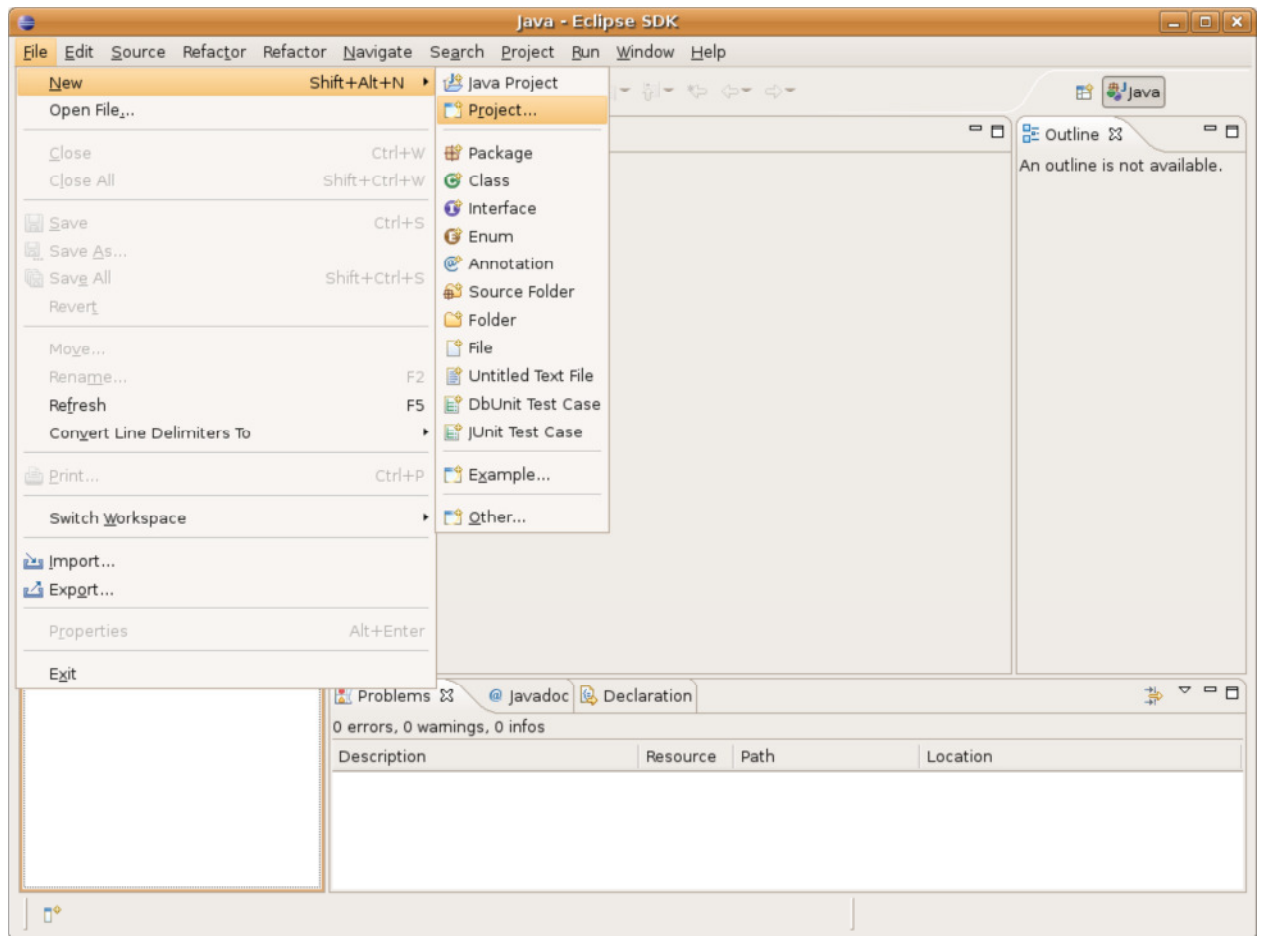


Figure 80: Create a new project in MOSKitt

Within the “MOSKitt” category there exists a special project called MOSKitt Project. Once it is selected, the next step is to give the project a name. To finish, it is necessary to click on the “Finish” button. A special perspective has been created which opens the explorer and positions it on the left. As this perspective is associated with the MOSKitt project type, Eclipse will ask the user if he/she wants to open this perspective. The work environment is as follows [33].

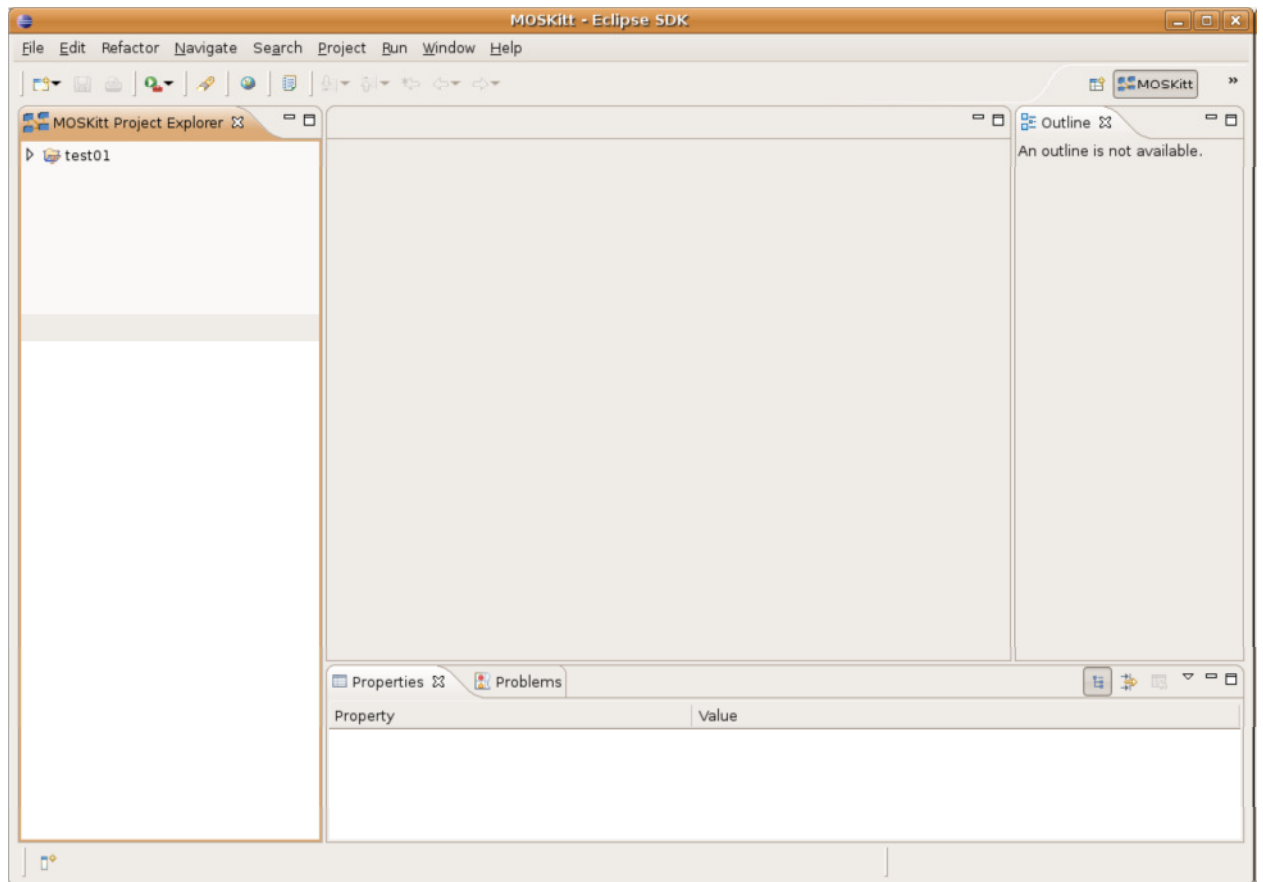


Figure 81: MOSKitt working environment

### 8.3 Openxava

OpenXava is a web application framework for developing business applications in an effective way. It not only allows rapid and easy development of CRUD (*Create, Read, Update and Delete*) modules and report generation, but also provides flexibility to develop complex real life business applications like accounting packages, customer relationship, invoicing, warehouse management, etc. OpenXava is a framework to develop JavaEE/J2EE applications quickly and easily. The underlying philosophy is to define with Java

annotations or XML and to program with Java; the more definition and less programming the better. The main goal is to build the most typical things in a business application easily, while still having the necessary flexibility to develop more advanced features as you require [47, 49].

The essence of OpenXava is that the developer defines instead of programming, and the framework automatically provides the user interface, the data access, the default behavior, etc. In this way, all common issues are solved easily, but the developer always has the possibility of manually programming any part of the application, in this way it is flexible enough to solve any particular cases. OpenXava is based on the concept of the business component [49].

A business component includes all software artifacts needed to define a business concept. OpenXava is a business component framework because it allows defining all information about a business concept in a single place. For example, for defining the concept of Invoice, in OpenXava a single file (Invoice.java) is used, and all information about invoice concept (including data structure, user interface layout, mapping with database, validations, calculations, etc.) is defined there. In an MVC (*Model View Controller*) framework the business logic (the Model), the user interface (the View) and the behavior (the Controller) are defined separately. These types of frameworks are useful if the rate of change of logic and data structures is low and the possibility of changing user interface technology or data access technology is high [47, 49].

Business components are the fundamental pieces to create applications in OpenXava. In the OpenXava context, a business component is a Java class (although there also exists an



XML version) that contains all the information about a business concept that is required to create applications. In a business component the user can define [47].

- The data structure.
- Validations, calculations and all logic associated with the business concept in general.
- The possible views of the component, i. e. the configuration of all possible user interfaces.
- The possibilities for the tabular data presentation. This is used in list mode (data navigation), reports, export to excel, etc. Object-relational mapping. This includes information about database tables and how to convert them to the objects of your Java application.

In OpenXava, the addition of a new field to an Invoice only requires changing a single file: Invoice.java. But MVC frameworks are cumbersome when changes to structure and data are very frequent (as in the business application case). Imagine the simplest change, adding a new field to an Invoice. In the MVC framework the developer must change three sections: the user interface, the model class and the database table Using OpenXava makes it possible to allocate the development work using a business logic oriented task distribution. For example Invoice to one developer, Delivery to another, as opposed to technology layer business logic to one developer, user interface to another [47, 49].

## **Model**

The model layer in an object oriented application contains the business logic, that is the structure of the data and all the calculations, validations and processes associated to this

data. OpenXava is a model oriented framework where the model is the most important, and the rest (e.g. user interface) depends on it.

The way to define the model in OpenXava is using plain Java classes (although a XML version is also available). OpenXava generates a full featured application from your model definition [47].

### **Business Component**

The basic unit to create an OpenXava application is the business component. A business component is defined using a Java class called *Entity*. This class is a regular EJB3 entity, or in other words, a POJOclass [54] with annotations that follows the Java Persistence API (JPA) standard. JPA is the Java standard for persistence, that is, for objects that store its state in a database. If you know how to develop using POJOs with JPA, you already know how to develop OpenXava applications. Using a simple Java class you can define a Business Component with [47]:

- **Model:** Data structure, validations, calculations, etc.
- **View:** How the model can be shown to the user.
- **Tabular data:** How the data of the component is displayed in list mode (in tabular format).
- **Object/relational mapping:** How to store and retrieve the object state from database [47].

### **Controllers**

A controller is a set of actions. An action is a button or link that a user can click on. The controllers define the things that user can do with the business component in the

application. These are specified in the file *xava/controllers.xml* of your project. In addition OpenXava has a set of predefined controllers in *OpenXava/xava/default-controllers.xml*.

The controllers are separated from the business components because one controller can be assigned to several business components. For example, a controller to make CRUD operations, to print in PDF format or to export to plain files, etc. can be used and reused for components like invoices, customers, suppliers, etc [47].

An OpenXava application is a set of modules. A module joins a business component with one or more controllers. Each module of the application is what the end user uses, and generally it is configured as a portlet within a portal. A typical OpenXava project contains these folders:

- **[root]:** In the root you can find *build.xml* (with the Ant task).
- **src[source folder]:** Contains your Java source code.
- **xava:** XML files to configure your OpenXava application. The main ones are *application.xml* and *controllers.xml*.
- **i18n:** Resource files with labels and messages in several languages.
- **properties[source folder]:** Property files to configure your application.
- **data:** Useful to hold the scripts to create the tables of your application, if needed.
- **web:** Web content. Usually JSP files, lib and classes. Most of the content is generated automatically, but you can put your own JSPs or other custom web resources here.

## 8.4 MOSKitt code generation model driven development

In the MDA approach to create an information system means to create the specific model that represents the domain under study. The first step taken in the MOSKitt approach is to create the UML model using the MOSKitt UML2 modeling module [33, 36]. MOSKitt approach uses only UML class diagrams [34].

Once the UML model is completed it is possible to create automatically the corresponding OpenXava application through model transformation. This is obtained through the UML2JPA automatic transformation process. The generated code will contain the standard Java types and the annotations for the classes, their properties and operations [34, 27].

Through this transformation process an OpenXava application with *CRUD* (*Create, Read, Update, Delete*) and *Export* (*Excel, PDF*) functionalities is generated that can be executed in the developing environment. The output OpenXava application is responsible automatically to generate a default user interface and ensure the persistence in the database. In case the default user interface is not satisfactory there is the possibility to define specific user interface layout. MOSKitt supports this through Sketcher model diagrams, which are user interface models, created by drag and dropping user interface widgets in the Sketcher editor [35]. In order to ensure consistence between the UML and Sketcher models every widget of the Sketcher model needs to be linked with the corresponding UML model property.

Let's describe how is possible to develop step by step an OpenXava application taking as origin only a UML2 model built using MOSKitt. First of all we are going to describe how to create the UML model named *invoice\_mode* [34].

### **Create the UML2 model with MOSKitt**

1. Select the option File/New.../MOSKitt UML Diagram.
2. In the next wizard page: Check the option Create new model and indicate that UML2 model must include all the UML2 primitive types by default (checking Add data types from Primitive Types):
3. Select as main diagram Class Diagram because we are going only to use these kind of UML2 diagrams:

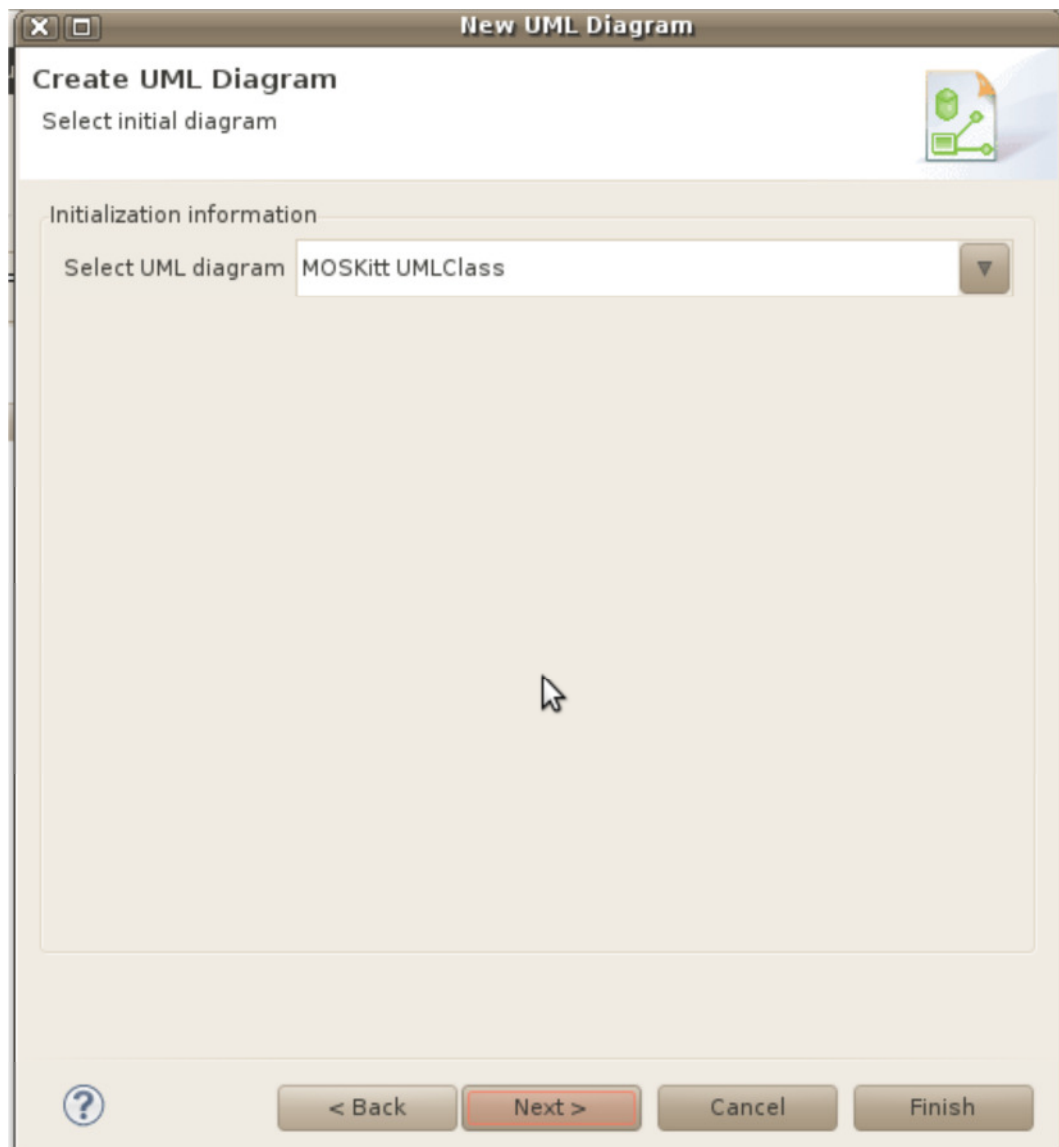


Figure 82: Create an UML2 model in MOSKitt

4. Select the project in which the diagram (.uml\_diagram file) must be included (Enter or select the parent folder) and give it a name (File name:):
5. Select the project in which the model (.uml file) must be included (Enter or select the parent folder) and give it a name (File name:). By default, MOSKitt will name

the model with the same name that the user selected for the diagram in the previous step:

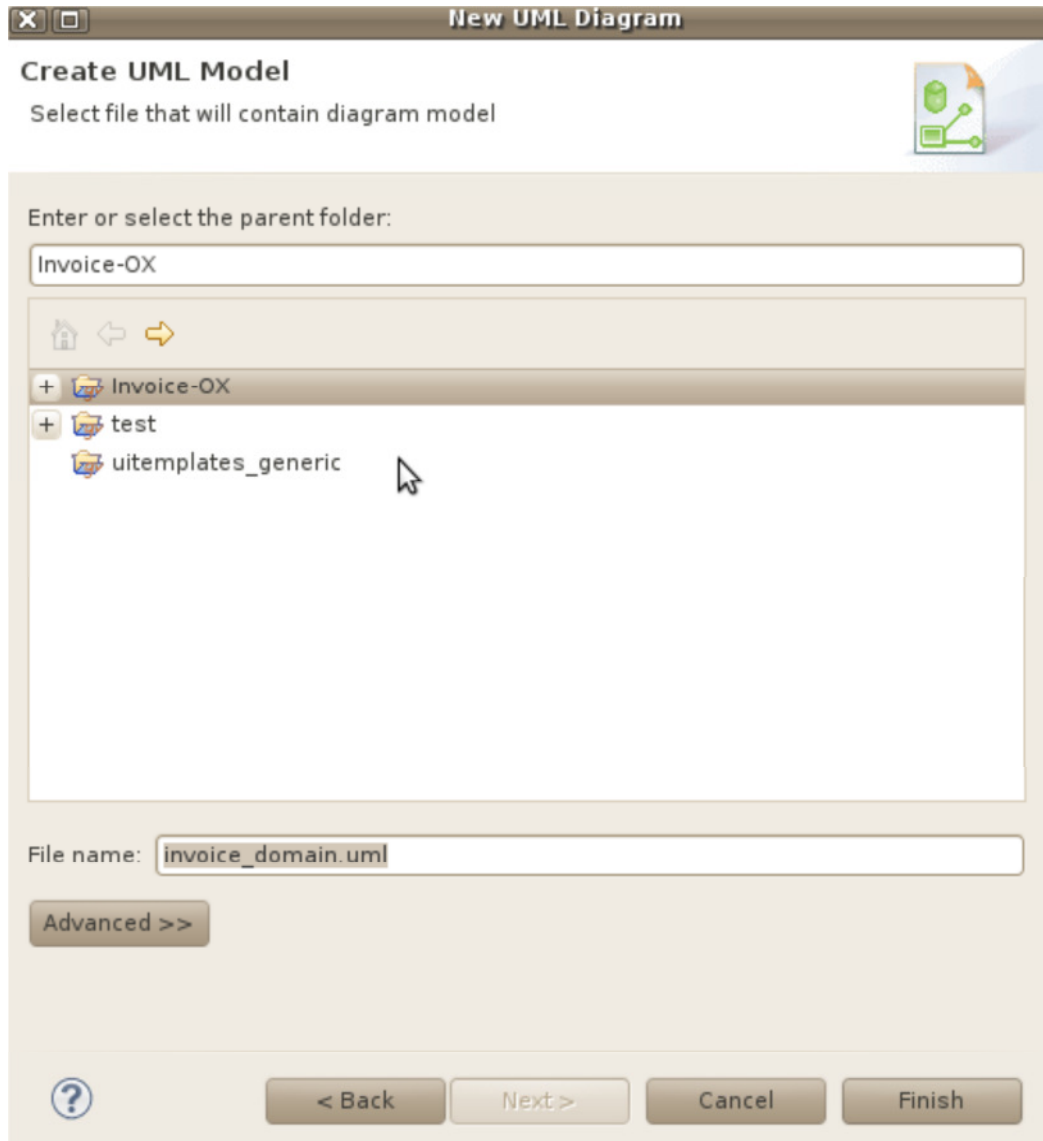


Figure 83: Create an UML2 model in MOSKitt, selecting the parent folder

6. Next press Finish and an empty class diagram will be opened to start to model the entities of the Information System. In the next figure you can see the Class Diagram

that we can use as example for modeling the business logic of an Invoicing Management application [34]:

The next step is to create an OpenXava project where locate the code generated by MOSKitt. This project must allow to be packed, deployed and executed in Liferay portal (included in the OpenXava distribution). To do this you must follow next steps [34]:



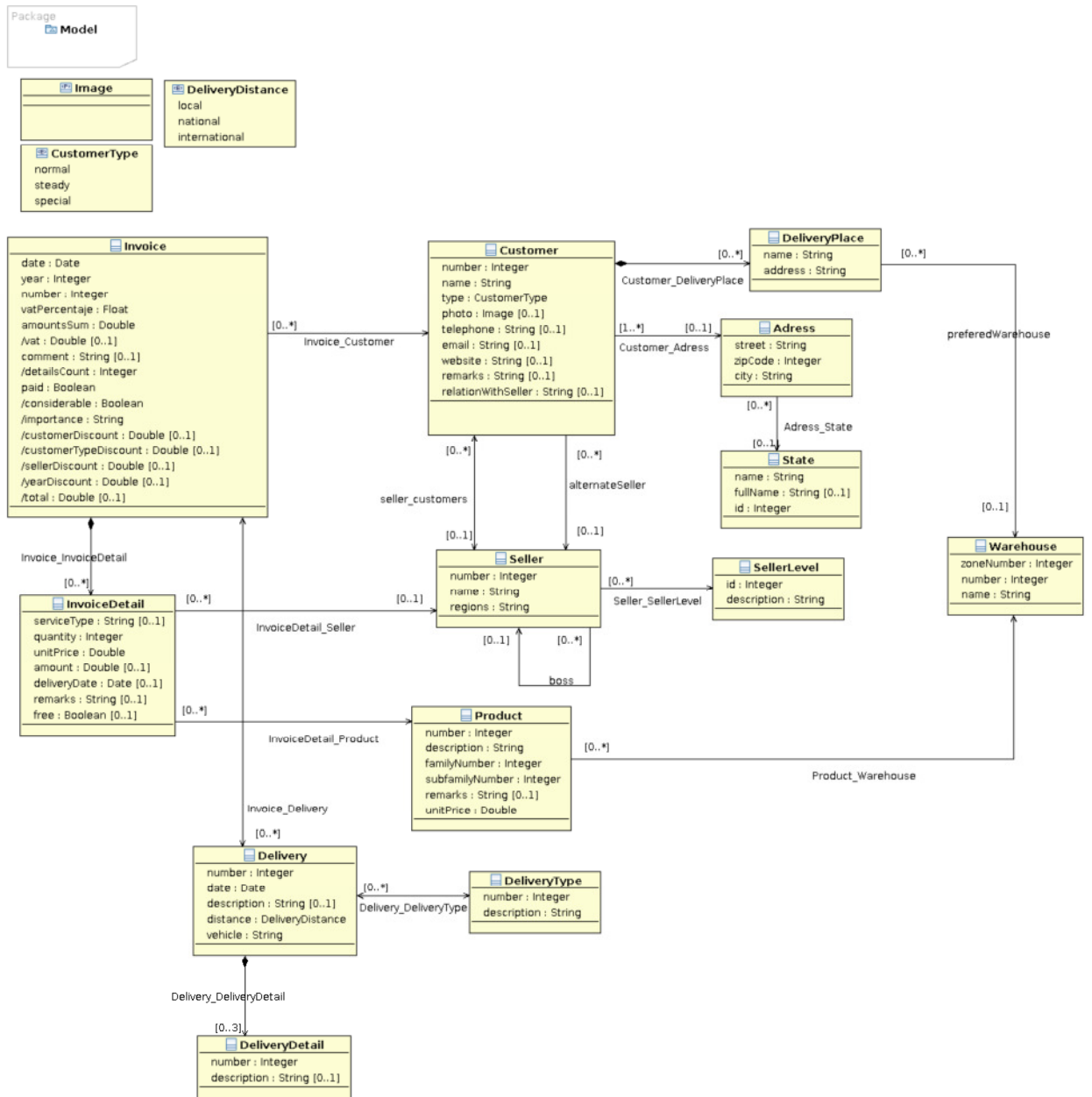


Figure 84: UML class diagram for the Invoice Management

## Generate OpenXava code from the UML2 model

1. Open MOSKitt selecting as workspace: *oxportal/workspace directory*.

2. It is recommended but not necessary to change to the *Java EE perspective* (*Window/Open Perspective/Java EE*).
3. Execute the menu action: *Run/External Tools/New OpenXava Project*.
4. Introduce the project name in the dialog window.
5. Although the OpenXava project has been created in our file system it is not accessible in the project view. To make it accessible you must follow next steps:
  - a. Execute the menu action: *File/Import...*
  - b. Select the option *Existing Projects into Workspace* and press *Next*.
  - c. For the option *Select root directory* select next path in workspace: *oxportal/workspace*.
  - d. The new OpenXava project will appear selected in the project list.
  - e. Press *Finish* and the OpenXava project will be available in the Project view.

The MOSKitt UML2JPA transformation allows generating Java code from elements in a UML2 model. The Java code generated will contain: The standard Java types declaration and The annotations for the classes, their properties and operations. These annotations allow to express and configure the generated code with specific elements of the JSR standard and OpenXava [34].

In the transformation process several resources take part [34]:

- Input Resources:
  - UML2 model (*invoice\_domain.uml*).
- Intermediate Resources:

- Configuration model of the UML2JPA transformation (*invoice\_domain.transformationconfiguration*). User can define in this model for each UML2 element which annotations must be applied.
- JPA model (*invoice\_domain.jpadeinitions*). This model completes the configuration model giving the required information for each selected annotations in the configuration model.
- Output Resources:
  - JavaGen model. It contains information about all the JPA and OpenXava annotations to be included in the Java classes (*invoiceJPA.javagen*).
  - Java classes with JPA and OpenXava annotations.

To launch the UML2JPA transformation the following steps are needed [34]:

1. Over UML2 model (*invoice\_domain.uml*) click the mouse right button to open the contextual menu, then select in the submenu *MOSKitt Transformations* the option called *UML2 to Java (JPA Persistence Entities) transformation*. To launch MOSKitt transformations user MOSKitt Perspective must be active (Window/Open Perspective/ MOSKitt).
2. Introduce the transformation parameters: the UML2 model, the folder where the JavaGen model must be located and the OpenXava project where the Java source code must be located by the transformation:

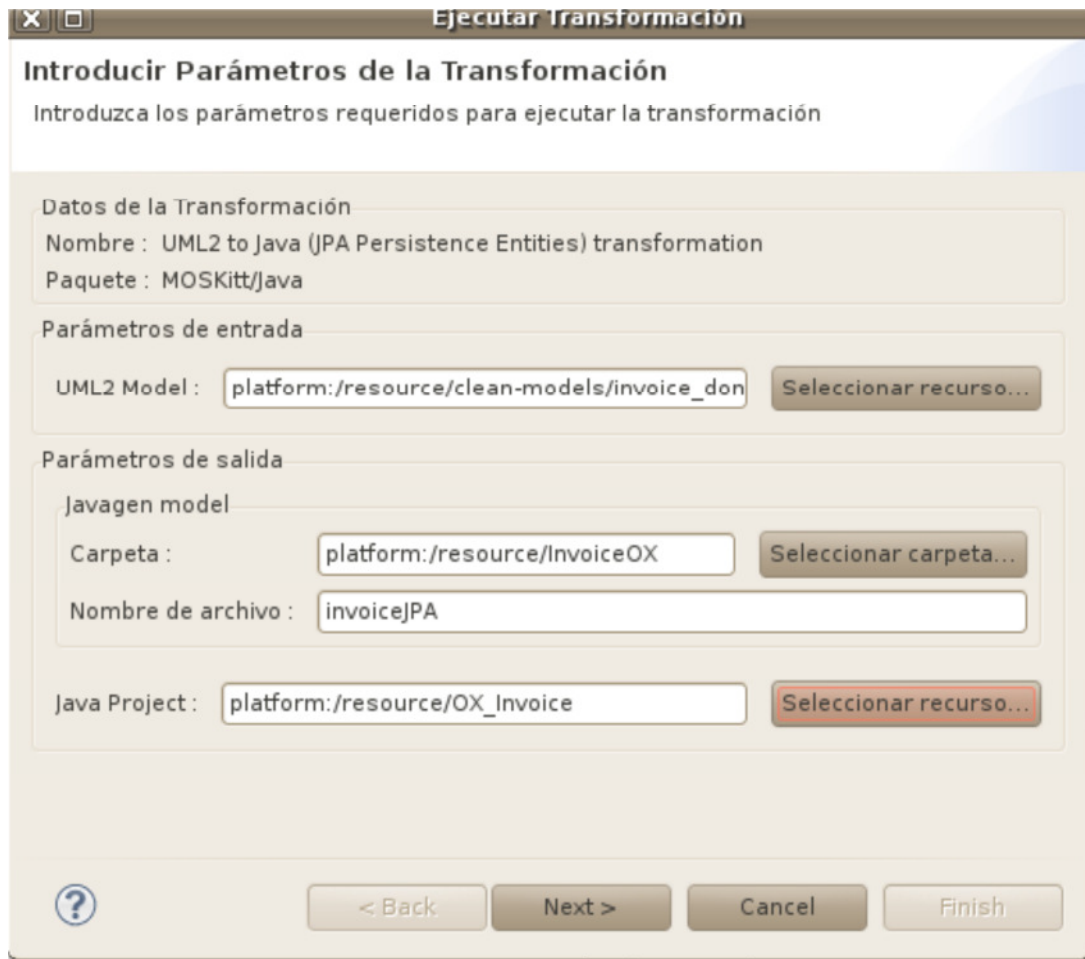


Figure 85: UML2JPA transformation, parameters introduction in the transformation

3. All MOSKitt transformation can be configured in order to alter the result depending on the features of each application. However, MOSKitt provides users with a default configuration. This default configuration applies the more commonly used transformation rules. In the example, we are going to use the default configuration (in later sections we can see how to modify this configuration) [34]:

4. Before launching the UML2JPA transformation a validation process checks if the input parameters are corrects. We mean, it checks if input models has the right type (in this case if it is an UML model) and also if their content is the expected content (when errors are detected the transformation only will be launched if all the errors are considered warnings) [34]:

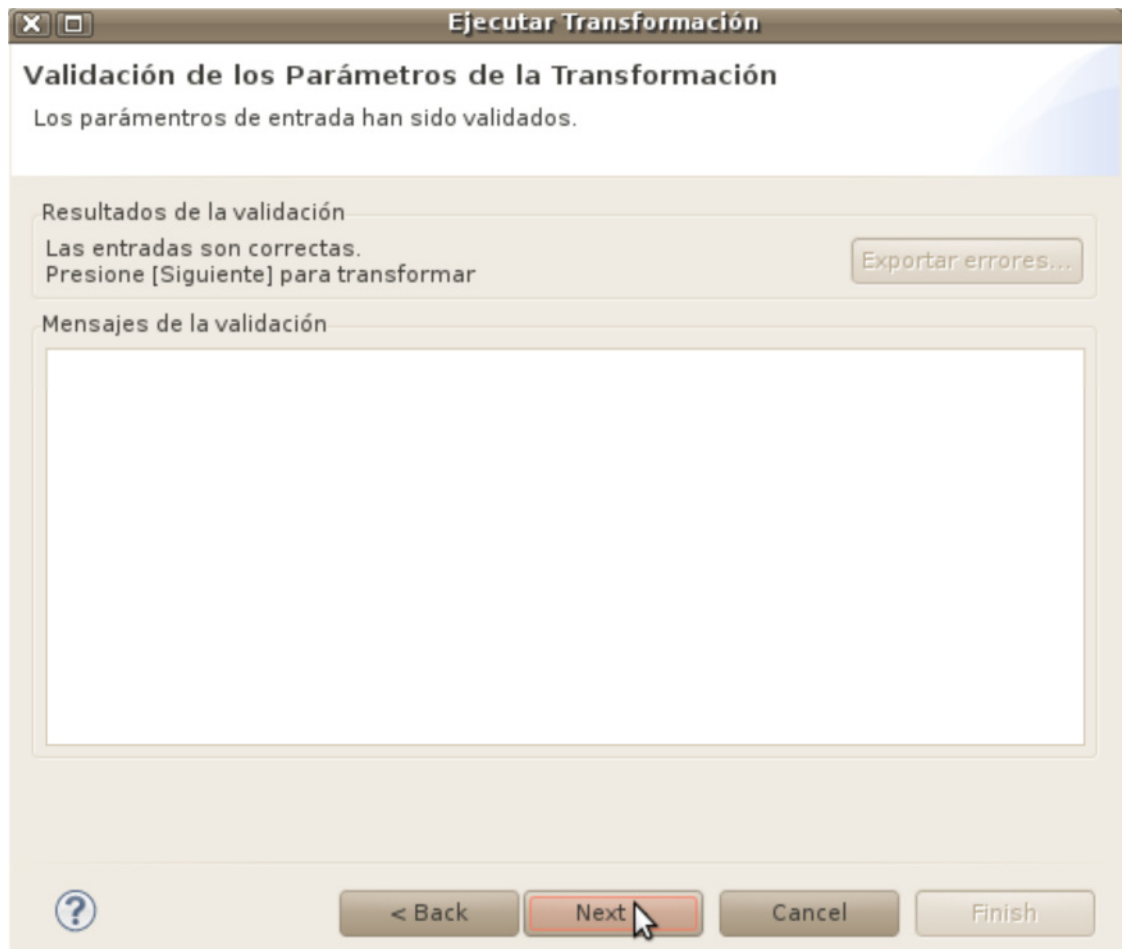


Figure 86: UML2JPA transformation, validating the parameters of the transformation

5. When transformation finishes a message will appear telling us that all was good [34].

From Java classes generated by MOSKitt, OpenXava applies its annotations to generate an AJAX JEE interface by default. You don't need to specify any User Interface model to obtain that. If Analyst needs to modify the User Interface design for some of the forms included in the application, he or she can include this kind of information using a User Interface Model provided by MOSKitt [34, 35].

## 8.5 OXPortal environment

The development environment is called OXportal, it deploys and executes the OpenXava configuration. Basically it's a compressed package with the following components [33]:

- **Apache Tomcat 6.0.29 Servlet/JSP container:** will put in execution the web applications that we deploy (including the portal itself)
  - Main location: oxportal/liferay/tomcat-6.0.29
  - Location of the start/stop scripts: oxportal/liferay/tomcat-6.0.29/bin
  - Location of the configuration files: oxportal/liferay/tomcat-6.0.29/conf
  - Location of the deployment folder: oxportal/liferay/tomcat-6.0.29/webapps
- **Portal JSR-286 Liferay 6.0.6 Community Edition:** will put in execution all the modules required by our application, the portlets for instance.
  - Main location: oxportal/liferay
  - Location of the deployment folder: oxportal/liferay/deploy

- HSQLDB (1.8.1) Database: will keep the data needed for the Liferay portal, it also will provide the storage for the database of our application.
  - JDBC driver location: oxportal/liferay/tomcat-6.0.29/lib/ext/hsqldb.jar
  - Liferay database location: oxportal/liferay/data
  - Database of our application location: oxportal/liferay/tomcat-6.0.29/data
- Workspace for MOSKitt with the OpenXava 4.2.x distribution: Environment that allows to develop OpenXava applications. If we connect this workspace through MOSKitt we will have all the necessary for generate and deploy our application in the portal.
  - Workspace location: oxportal/workspace

For getting ready our environment we only need to extract the oxportal-4.2.x.zip archive in any directory [33].

### **OXPortal environment installation**

Now we have our OpenXava project in our workspace ready for be completed. The next step is to create a HSQLDB database instance with the corresponding tables and configure the project for access to the database. For creating the database instance we must follow the next steps [33]:

1. Open a shell terminal.
2. Go to the oxportal/liferay/tomcat-6.0.29/bin directory.

3. Execute the HSQLSB database instance creation order: `./start-hsqldb.sh`  
`<instance_name> <port>`. We must consider a name and a port for our instance, for example: `./start-hsqldb.sh invoicing 9001`
4. this moment we have an empty database instance that can be accessed through JDBC using this URL: `jdbc:hsqldb:hsq://localhost:<port>/<database instance>`

Other mandatory accessing parameters are:

username: sa

password:

For configuring the database access from MOSKitt for creating and updating the database scheme tables we will follow the next steps [33]:

1. Edit the file `<mi_project>/persistence/META-INF/persistence.xml`
2. Find the "`<persistence-unit>`" called "junit". We must complete the property "`hibernate.connection.url`". Specify the JDBC url to our database:  
`jdbc:hsqldb:hsq://localhost:<puerto>/<nombre_instancia_db>`
3. 3. Save the file.

In order to allow the liferay portal the access to the database we will follow the next steps [34]:

1. Verify that the Tomcat server is stopped. If it does not you must stop it  
(`./shutdown.sh`)
2. Edit the file: `oxportal/liferay/tomcat-6.0.29/conf/context.xml`
3. Add a `<Context>` tag to the next entry:



```

<Resource name="jdbc/<my_project>DS" auth="Container"
type="javax.sql.DataSource"

maxActive="20" maxIdle="5" maxWait="10000" username="sa" password=""

driverClassName="org.hsqldb.jdbcDriver" url="jdbc:hsqldb:hsq://localhost:<port>/

<db_instance>"/>

```

4. With the Tomcat server up and running, the applications deployed in it will have available a JNDI resource named "jdbc/<my\_priject>DS" which we have registered previously in the global context of the container. That resource is a DataSource that will be used in our application in order to acces the database [33].

### **Creation of the database tables from JPA entities of an OpenXava project.**

Once we have the code of the JPA entities in our application (generated by MOSKitt or not) is the moment for creating the tables in the database. We must consider that the database instance must be running as we explained in the last point. From MOSKitt follow the next steps;

1. Open the file <my\_project>/build.xml
2. In the "Outline" view we will see a list of available targets for Ant. Choose a target named "updateSchema". Open the contextual menu and click <Run As>,<Ant Build>
3. In the "Console" view we will see an execution trace of the selected script. In that view we will see how the tables are created and if the process finishes well

or not. The next step is to inspect the tables. For doing it we will use a database manager compatible with JDBC (for example, SquirrelSQL).

### **Packing and deployment of an OpenXava project as portlets.**

For deploying an OpenXava application previously generated we have two options:

- Packing and deployment as a standard web application (WAR): with this option we can pack the application as a .WAR (Web Application Archive). This archive can be deployed in any standard Servlet container, for example Apache Tomcat, Jetty, etc. The result is that each module is accessible through a URL, perhaps that situation is not very comfortable for the user [33].
- Packing and deployment as a Web application with portlets (JSR-286): Allows us to deploy the application in a portal that supports Portlets 2.0 (JSR-286) standard. Each module of the application is transformed in a portlet. The portal allows us to configure the access to all the portlets establishing access policies to the modules for the portal users. Also allows us to use components for improving the accessibility (menus) of the user. This is the way used for the deployment and execution of the application in the example[33].

We will see how to do it with the second way, so we will use deploy our Web application with portlets: Now that we have our database available and the portal in execution, our application is ready to deployment and execution. In the file build.xml inside our project we have a target named "deployPortlets" that does all the tasks just once. This tasks are [33]:

- Compiling of the source code of the application.

- Generating the needed descriptor files: web.xml y portlet.xml
- Packaging of the application as a .WAR file.
- Deployment of the application in the Liferay portal [33].

From MOSKitt follow the next steps [33]:

1. Open the file <my\_project>/build.xml
2. In the "Outline" view we will see an available target list for Ant. Choose a target named "deployPortlets", Open the contextual menu from it and click <Run As>. <Ant Build>.
3. In the "Console" view we will see an execution trace of the selected script. In it we will see how it compiles, packs and copies to the Portlet application deployment folder in the portal.

If we have a terminal opened showing the output messages of the Tomcat container as is suggested in the last point, we will see the process of the application deployment in the Liferay portal. That will help us to detect errors in the deployment and to see how our application is ready for be used [33].

### **Access to the application in execution from the Liferay portal.**

Now we have deployed our application in the portal. The next step is just accessing to the portal and configure it for enabling the access to the portlets to the end users. The Liferay instance distributed with the oxportal-4.2.x.zip package comes configured with an admin user with enough privileges for creating pages that will allow us to execute the portlets or

modules of our application. We will create a page in the portal in order to gain access to the portlets of our application. For doing that we must follow the next steps [33].

1. Open the explorer and point the URL to <http://localhost:8080>
2. In the Login portlet put the following admin credentials:
  - a. User: [test@moskitt.org](mailto:test@moskitt.org)
  - b. Password: test
3. From the top menu, execute <Manage>.<Control Panel>
4. In the left panel in "MOSKitt Web Application Generation" section, click on the option "Pages".
5. In the current form we are going to create a new page. For doing that we will put the name of the page and select the "Panel" type. After that click the button "Add page".
6. With the page created, it's time to configure it. Select the page in the menu, after that, click the tabular option named "Page". Then, a form will appear showing the configuration of the page.
7. In the configuration form of the page we can see a section named "Select the portlets that will be available in the panel". In that section we must have our application available with all their portlets. Select the portlets you will need for being accesibles by the final user. After that, click "Save".
8. Our page is configured, for accessing the application from the top menu. Click the option "Return to MOSKitt Web Application Generation". Now we will see one new option with the name of our page. If we click on it, a portlet with the

list of modules in our application will appear in the left menu. Clicking in any of the modules will show on the right side of the layout, the execution of the corresponding module.

## **CHAPTER IX: PRACTICAL APPLICATION IN THE ENTERPRISE**

### **9.1 The business process**

In big and multinational companies that need to comply with international standards like ISO27001 there needs to be in place a process to control the access of the employees into systems (CRM, ERP etc) and also document the requests of the process for auditing purposes. This process is commonly known as the user access management process, and is the process which we are taking into consideration. In order for an employee to gain access into a system the employee needs to perform a request specifying in it the details of the access needed, the system when access is needed, the period for which the access is needed, priority of the request and other useful information. The request goes through an approval cycle which includes the line manager of the employee and the Corporate Security manager. Corporate Security manager is an entity within the company which is responsible for the logical and physical security and all accesses into systems go through his/her approval. If the request is rejected from the line manager and/or Corporate Security manager the process ends and the employee is informed. Only when the request is approved by both the line manager and Corporate Security manager it goes for implementation to the respective administrator. In the following sections we will automate the above process using both model driven architecture approaches described above, Bizagi and MOSKitt framework.

### **9.2 Automating the process in Bizagi**

As per the model driven architecture principles to create information system means to create the specific model that represents the domain under study. In Bizagi process management suite the first step is to create the BPMN model using the process modeler [1, 9]. The BPMN model created for the User Access Management flow is represented in Figure 1, containing all the stakeholders involved in the process with the respective tasks/activities that each should perform. The start and end event are also easily distinguished [16, 58]. The model is straightforward. The process starts when an employee enters a permission request for a particular system within the company. After the request has been completed from the employee it goes for approval to the respective line manager. If the line manager rejects the request, the employee is informed for the rejection reason and the process ends. On the other hand if the line manager approves the request flows to the Corporate Security Manager, which is a central entity within the company responsible for the security of the systems and which approve all permission requests. The Corporate Security manager also makes a choice. In case the request is rejected the employee is notified and the process ends. Only if the request is approved also from the Corporate Security manager it goes for implementation to the respective implementation team within the company. After the request is implemented the Corporate Security analyst is informed and the process ends. The employee is being granted with access in the system asked.

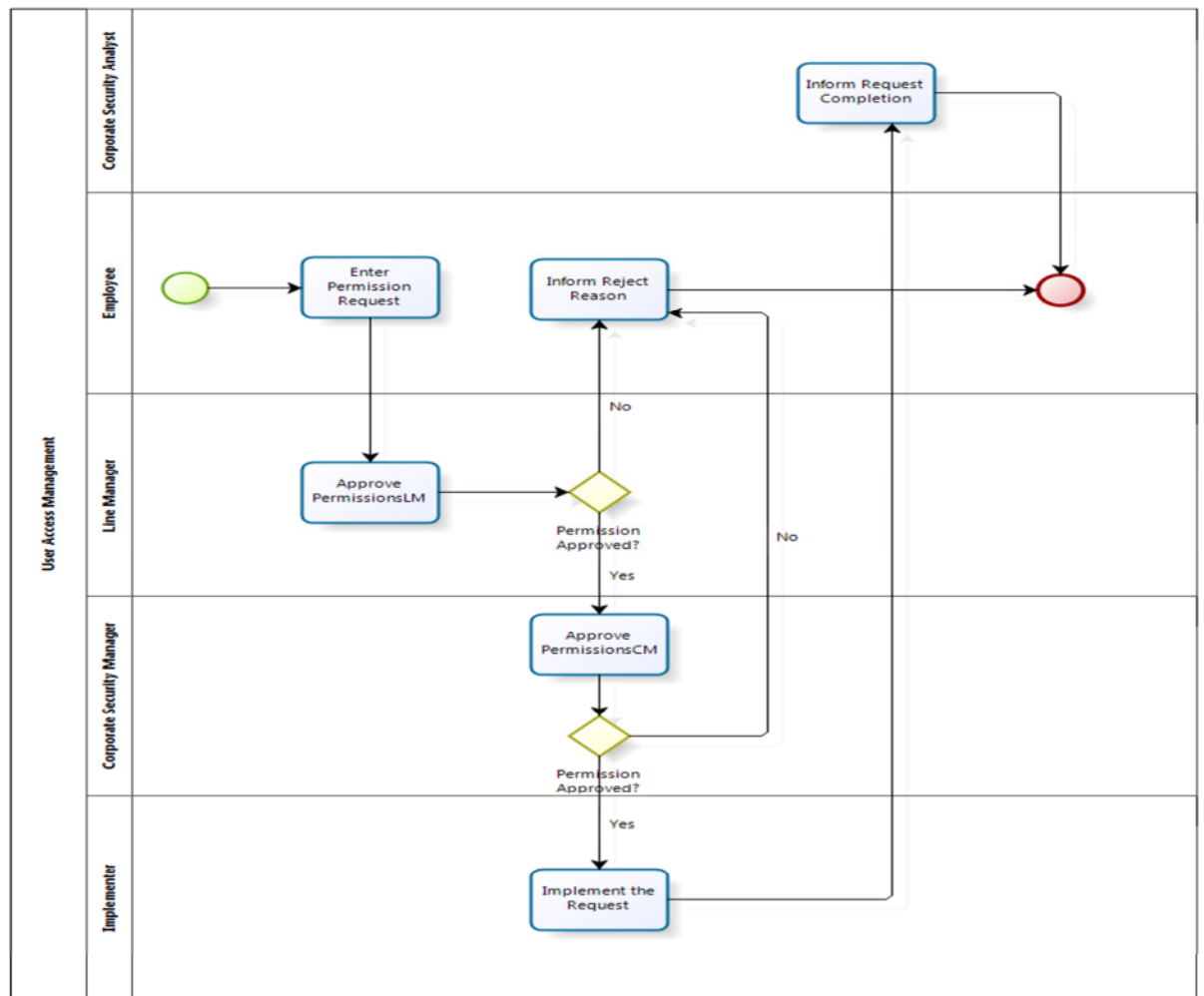


Figure 87: BPMN model for the user access management flow

Once the BPMN model is completed it is possible to create automatically the running application by following the process automation steps in Bizagi Studio [1]. The process automation wizard in Bizagi Studio is comprised of several steps that are necessary to enrich and automate the BPMN process produced from the process modeler as shown in Figure 24.



## Data Modeling

The first step is to define the data that the process requires for its execution. Bizagi allows to structure business data in a graphical and logical way similar to an Entity-Relationship schema, resulting in an easy to understand Data Model. To provide an organized and coherent structure Bizagi provides different types for Entities and different types of Relationships that can be used in order to build the data model. Entities are real or abstract objects (people, places, events, and so on) that can be uniquely identified and are of interest to the business about which information is stored in the system. Entities can be usually thought of as nouns. For example *a Customer, a City, a Company, an Invoice, a Car*. Entities have Attributes. These are the properties that describe each entity. For example a customer has a name, a social security number a gender and an age [1].

Every process in Bizagi has its main Process Entity. This entity provides the starting point to access the rest of the process data, that is, it is the principal entity through which a user accesses the rest of the data model entities. Relationships capture how entities relate to one another and can be thought of as verbs. For example: *a Customer owns a car, a company is located in a city, a customer has a gender*. In Figure 88 is presented the data model created for the User Access Management flow containing all the data needed during the process execution.

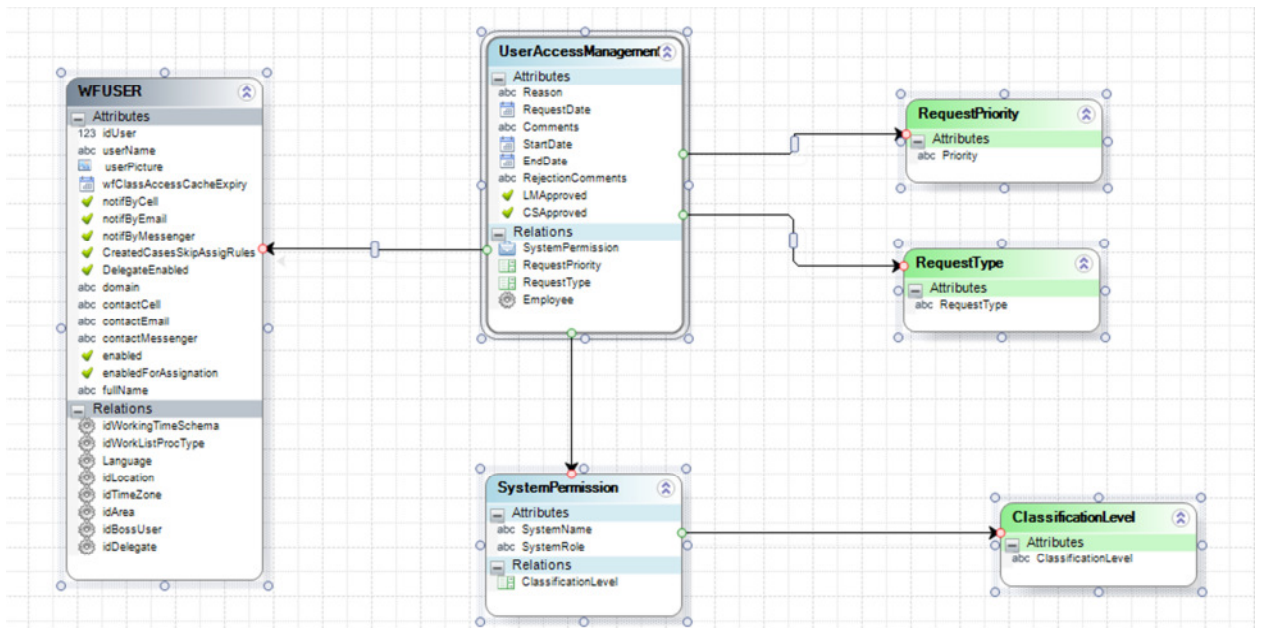


Figure 88: The data model for the User Access Management flow.

Bizagi Studio provides ready for use a set of entities which belong to the Bizagi's internal data model. These are called System entities and the **WF USER** entity presented in Figure 88 is one of them. System entities include information concerning the end users, areas, locations etc. These entities are created by default for each project and help in the creation of the data model for the specific process. All is needed is to include and link these entities in the data model created. They represent general entities which are present in any process and thus help to focus our attention in creating the entities which are particular for our process and utilizing System entities for general use cases (like end users in Figure 88).

Creating the data model is a continuous process that goes throughout all the steps of process automation until the moment that we are convinced that all the information needed throughout the process execution is present in the data model.

## Forms – User Interface

The second step is to define the user interface for all the activities in the model that require human interaction. In the *User Access Management* process all the activities require human interaction since there are not present automatic activities. To define the user interface for an activity the **Forms Designer** is used which provides an intuitive and user friendly structure to drag and drop the data fields onto a form and arrange them accordingly without the need of programing [1]. In Figure 4 is presented the user interface created for the first activity of the *User Access Management* process, *Enter Permission Request*.

Figure 89: User interface for the Enter Permission Request activity.

Each item or element included in the design area is known as a *Control* in Bizagi. *Controls* can be added individually by means of drag and drop from the left panel. When a *Control* is added in the design area it does not have a reference in the data model to the specific

attribute it represents. In order to link the *Control* with the corresponding attribute in the data model the *Data Source* property of the *Control* should refer the correct attribute in the data model. In Figure 89 is shown how the *Data Source* property of the *StartDate* control is related with the *StartDate* attribute in the data model

(*UserAccessManagement.StartDate*). Following the same process is possible to create the user interface also for the remaining activities in the process [1].

## **Business Rules**

Bizagi provides powerful business rules that can be implemented in various places throughout the process. The following points clarify the use of business rules.

*Perform actions in Activities - calculate and validate:* It is very common to perform validations to control the values entered by end users in a field. Bizagi also allows making calculations to manage the data model and assign or delete values to any attribute.

*Route Processes - Sequence flow:* Business rules are implemented as transition conditions in a process to route the sequence flow. The expressions used for this purpose always return a value of **True** or **False** (Boolean) in order to tell the process where to go next.

*Managing the user interface:* it is very important to be able to control what fields are displayed or hidden to the end users, as well as control what fields are mandatory, editable or read-only.

*Users allocation:* Business rules can be used to control the users that will be allocated to tasks according the business conditions [1].

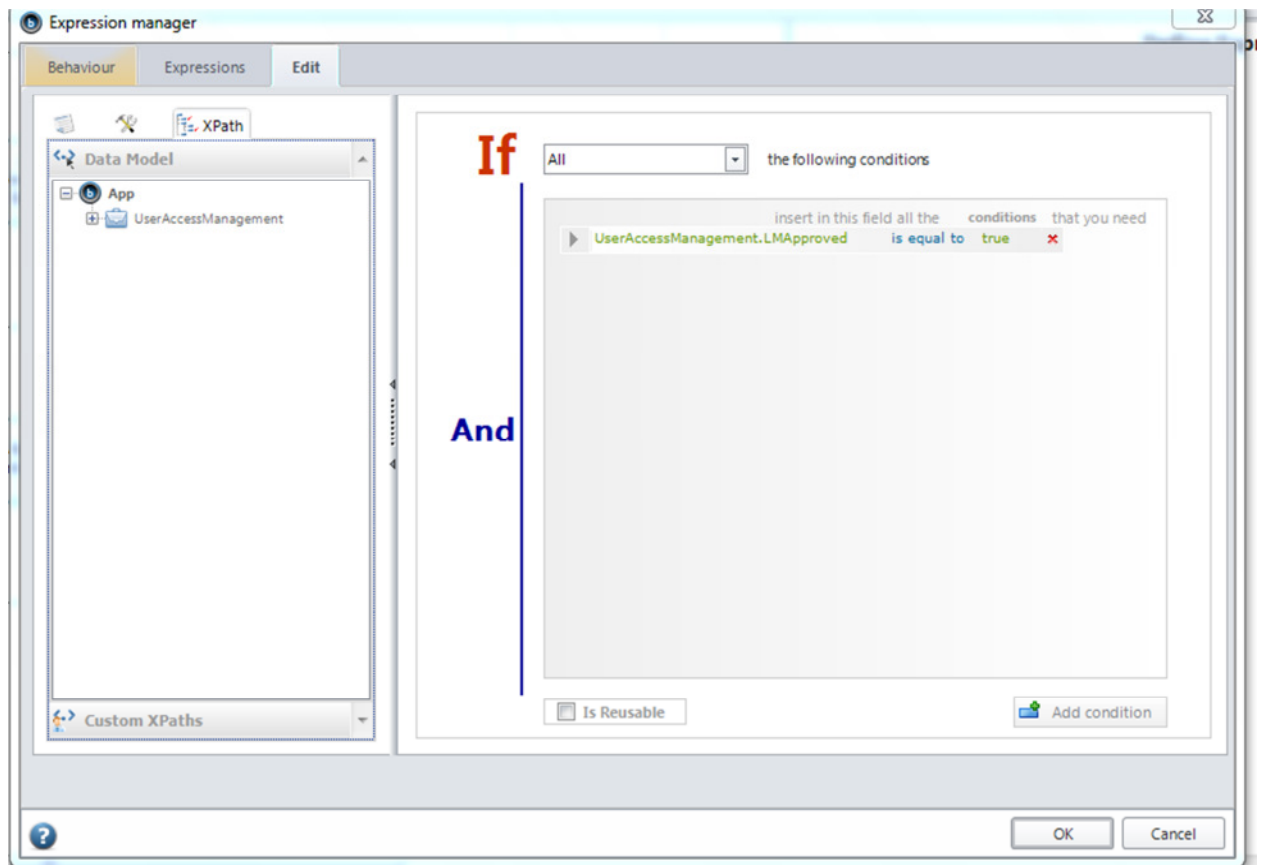


Figure 90: Business rules to route the User Access Management process

In Figure 90 is presented how business rules are used in order to route the user access management process. Let's consider the case when the Line Manager has approved the request. This is represented by the following expression

*(UserAccessManagement.LMApproved is equal to true)*. LMApproved is a boolean attribute present in the data model as shown in Figure 88 that will store the choice performed from the Line Manager. In a similar way are defined the business rules for the other routes of the process.

### Define performers

Work allocation is the next step of the process automation wizard where *Performers* are defined for each activity of the process. *Performers* are the users that have the qualities to be assigned to activities. Each activity created for end user interaction requires definition that will allow Bizagi to allocate the correct users within the organization. Bizagi automatically evaluates the allocation rules defined for each activity and selects one or more users that meet the given conditions from the user's list. Only these users will have access to work on the activity allocated to them. To allocate performers, it is necessary to have a user account created for everyone that intends to work with Bizagi. It is important that each user account must be set up correctly to ensure Bizagi selects appropriately. This powerful feature can be configured based on different criteria like positions, geographical location, skills, roles, among others.

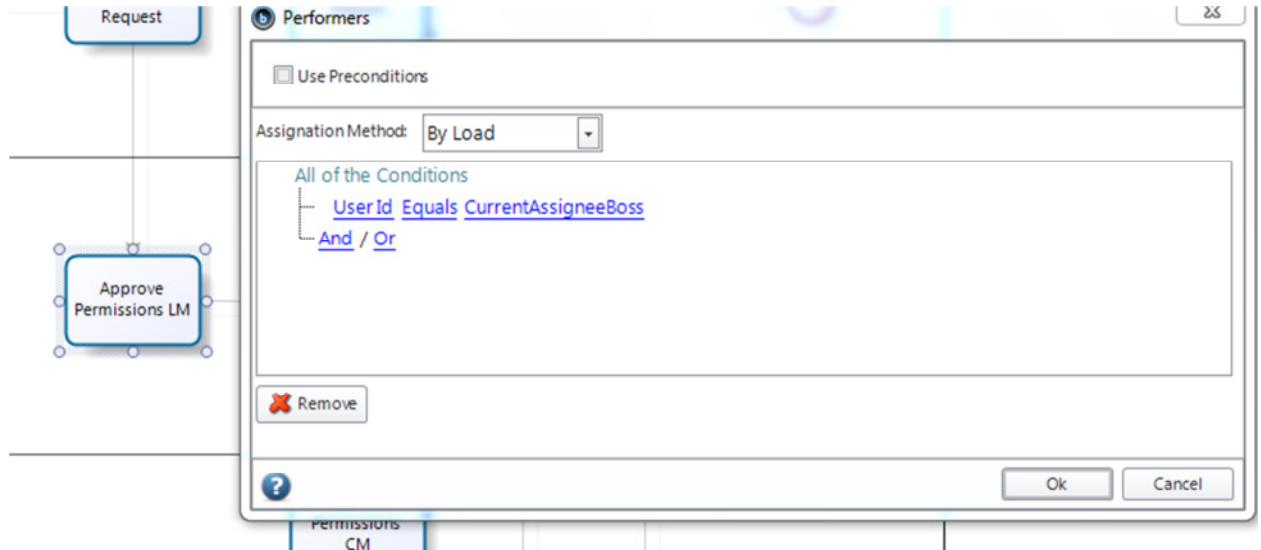


Figure 91: Define performers for the activity Approve Permissions LM

In Figure 91 is shown how is defined that the activity *Approve PermissionLM* is always performed from the line manager of the user that has raised the request. The expression used is ***UserId Equals CurrentAssigneeBoss*** that instructs Bizagi that the activity *Approve Permissions LM* should be performed from the line manager of the user that submitted the request in the first place. Similarly in Figure 92 is shown how the activity *Approve Permission CM* is allocated to the Corporate Security Manager through the application of positions present within the company [1].

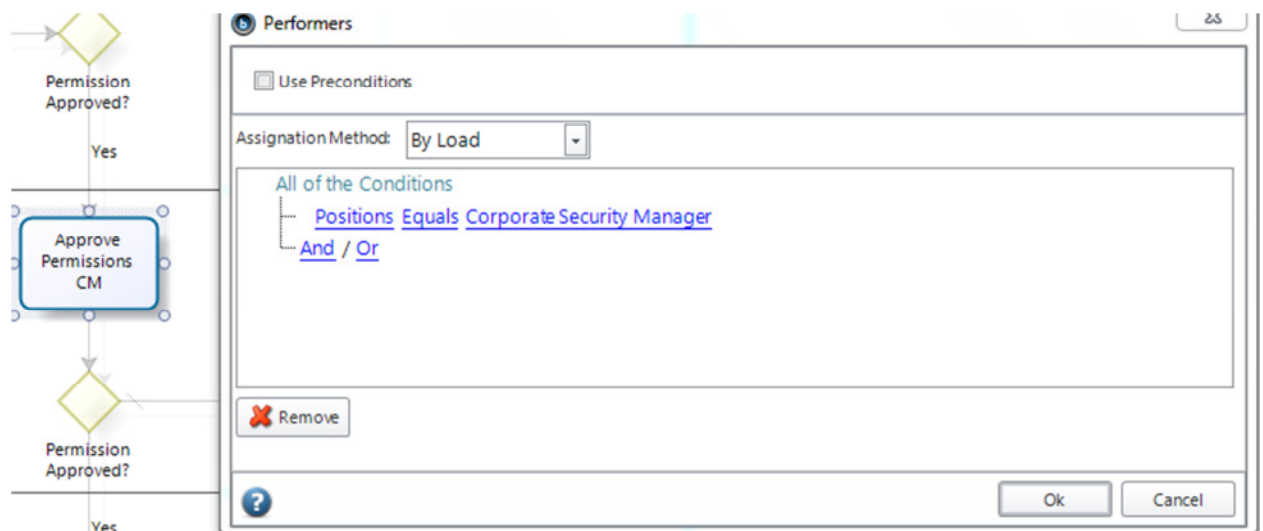


Figure 92: Allocation of the Corporate Security manager to the activity Approve Permissions CM.

## Integrations

The next step in the automation wizard provides the capability to integrate the process with other external systems present within the enterprise. In the *User Access Management* process there are considered no integration with external systems.

## Execution

Once the process has performed all the steps of the process automation it is ready to be executed in the Bizagi execution environment, in Bizagi Engine. An enterprise web application is generated automatically which is ready for execution starting from the BPMN model without the need of programming. The user access management application in execution is presented in Figure 93.

The screenshot displays a web application interface for 'Access Request Information'. The browser's address bar indicates the URL 'localhost:1024/AccessManagement/Default.aspx'. The Bizagi logo is visible in the top left corner. The page features a search bar and a navigation menu with icons for adding, checking, reporting, and settings. The user is logged in as 'admon' and can click 'Logout'. The main form area contains the following fields:

- Employee: (text input)
- RequestDate: 9/27/2013 (text input)
- Reason: (text input)
- Comments: (text input)
- RequestType: (dropdown menu)
- RequestPriority: (dropdown menu)
- StartDate: (text input)
- EndDate: (text input)
- SystemName: (text input)
- SystemRole: (text input)
- ClassificationLevel: (dropdown menu)

At the bottom of the form, there are 'Save' and 'Next' buttons.

Figure 93: User Access Management application in execution

### 9.2.1 Limitations

Starting from a BPMN model that represents a particular business process and following the process automation wizard in Bizagi Studio is possible to generate a web application that automates the business process without the need to write any code. Bizagi business



process management suite fully supports the MDA principles. Although the following points need to be taken into consideration as limitations:

- a) Bizagi is a suite for process automation and is not applicable for general purpose application development.
- b) The processes automated with Bizagi can be executed only within the Bizagi environment (Bizagi Engine).
- c) Bizagi is a proprietary solution since respective license fees need to be paid for enterprise deployments.

### **9.3 Automating the process in MOSKitt**

The first step taken in the MOSKitt approach is to create the UML model using the MOSKitt UML2 modeling module [33, 36]. In Figure 94 is shown the UML model created, an UML class diagram, containing the classes with their attributes and relationships between them for all concepts necessary to cover functionalities required to deal with the UAM process. MOSKitt approach uses only UML class diagrams. The two most typical types of request are considered (grant access, revoke access). The model is straightforward. Companies are composed of several departments and have only one main address. An employee working for a company performs a request (grant access or revoke access) in a specific date to obtain access for a particular system, specifying the access period needed and the priority. The role group of the system to which access is required with the respective roles is specified. A predefined approval path is associated with each request. Upon completion of the approval the request is implemented.

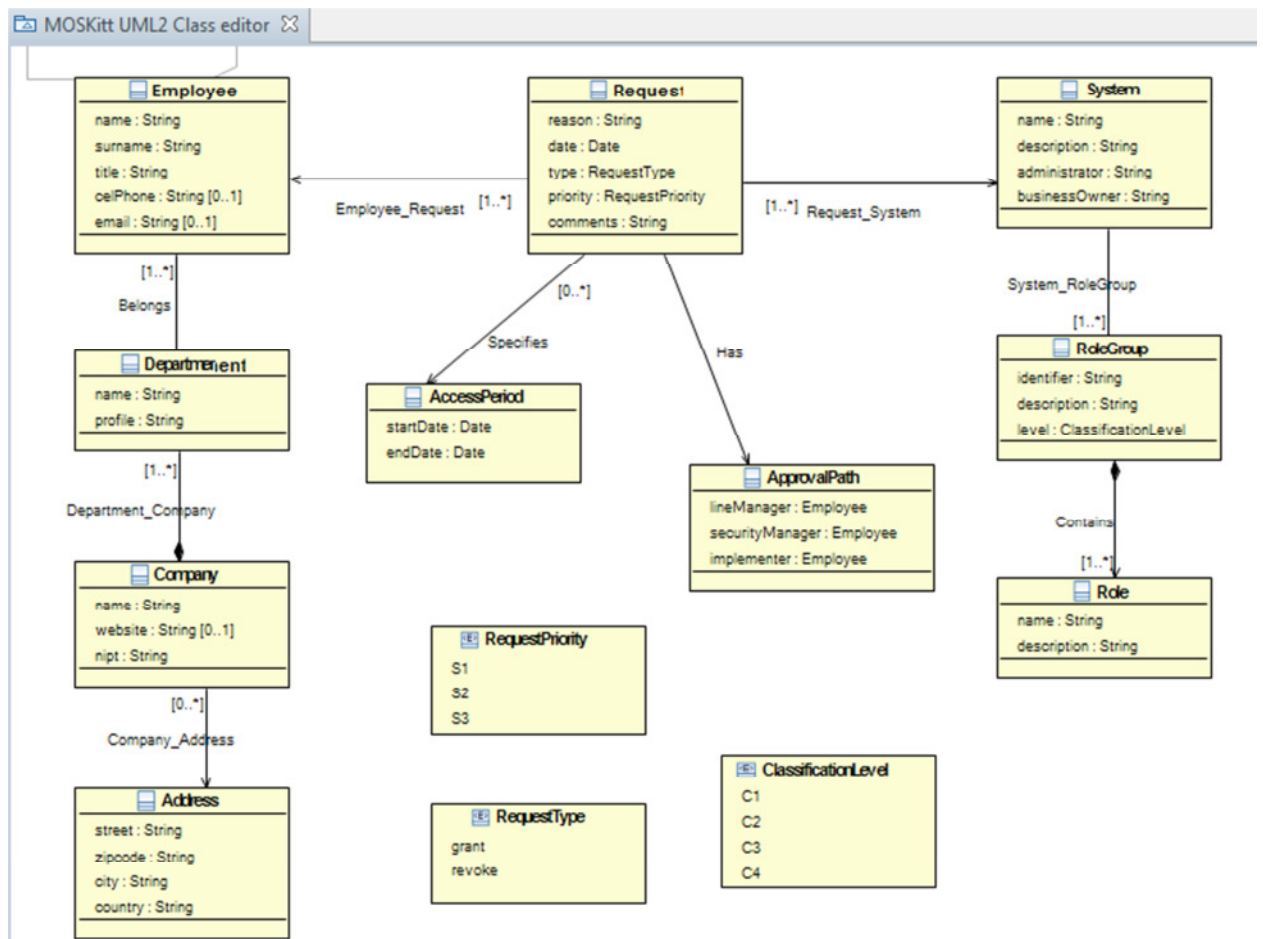


Figure 94: UML model of the User Access Management flow.

Once the UML model is completed it is possible to create automatically the corresponding OpenXava application through model transformation. This is obtained through the UML2JPA automatic transformation process. The generated code will contain the standard Java types and the annotations for the classes, their properties and operations [34, 27]. Through this transformation process an OpenXava application with *CRUD* (*Create, Read, Update, Delete*) and *Export* (*Excel, PDF*) functionalities is generated that can be executed

in the developing environment. The output OpenXava application is responsible automatically to generate a default user interface and ensure the persistence in the database. In case the default user interface is not satisfactory there is the possibility to define specific user interface layout. MOSKitt supports this through Sketcher model diagrams, which are user interface models, created by drag and dropping user interface widgets in the Sketcher editor [35]. In order to ensure consistence between the UML and Sketcher models every widget of the Sketcher model needs to be linked with the corresponding UML model property.

The model driven development adopted from MOSKitt is a chain of model transformations as presented in Figure 95 [34].

**Step 1:** Generate the Java classes with the JPA annotations for the persistence of the UML entities. This is obtained from UML2JPA transformation from the UML Model.

**Step 2 [Optional]:** Add the OpenXava annotations to the Java classes to express the user interface information specified by means of the Sketcher model. First the Sketcher model is transformed in the User Interface Model (Sketcher2UIM) and the latter through UIM2OX transformation provides the final OpenXava code.

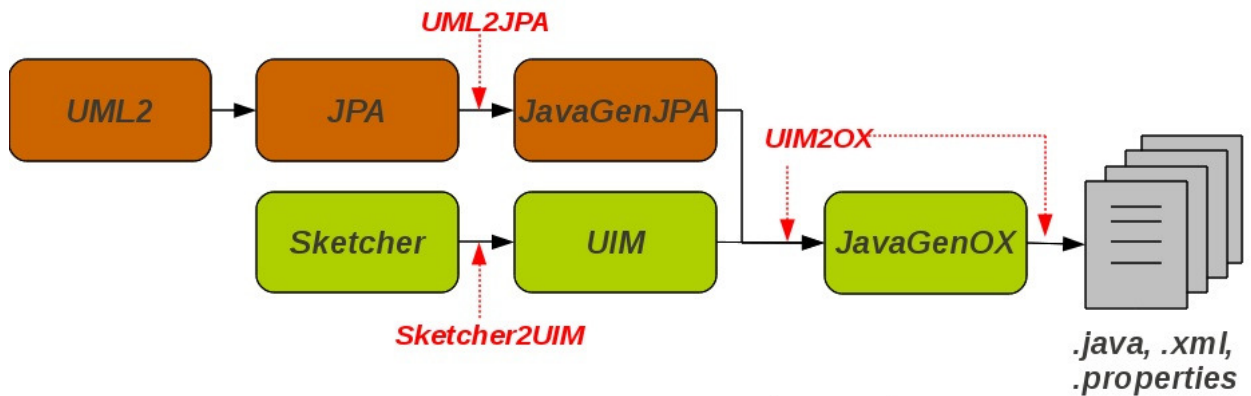


Figure 95: MOSKitt model driven development strategy – chain of model transformations.

In Figure 96 is shown the Sketcher Diagram for the Request module, presenting the list view and part of the detailed view for the module. The “**Name**” widget that will contain the name of the employee performing the request is mapped with the “name” property of the UML Model, “**name**” property of the class **Employee**. To ensure this the attributes *Data Model Element* and *Data Model Path* are set accordingly [35].

**Data Model Element:** <Class> Employee :: <Property> name : String

**Data Model Path:** <Class> Request :: <Association> Employee\_Request :: <Class>

Employee :: <Property> name : String

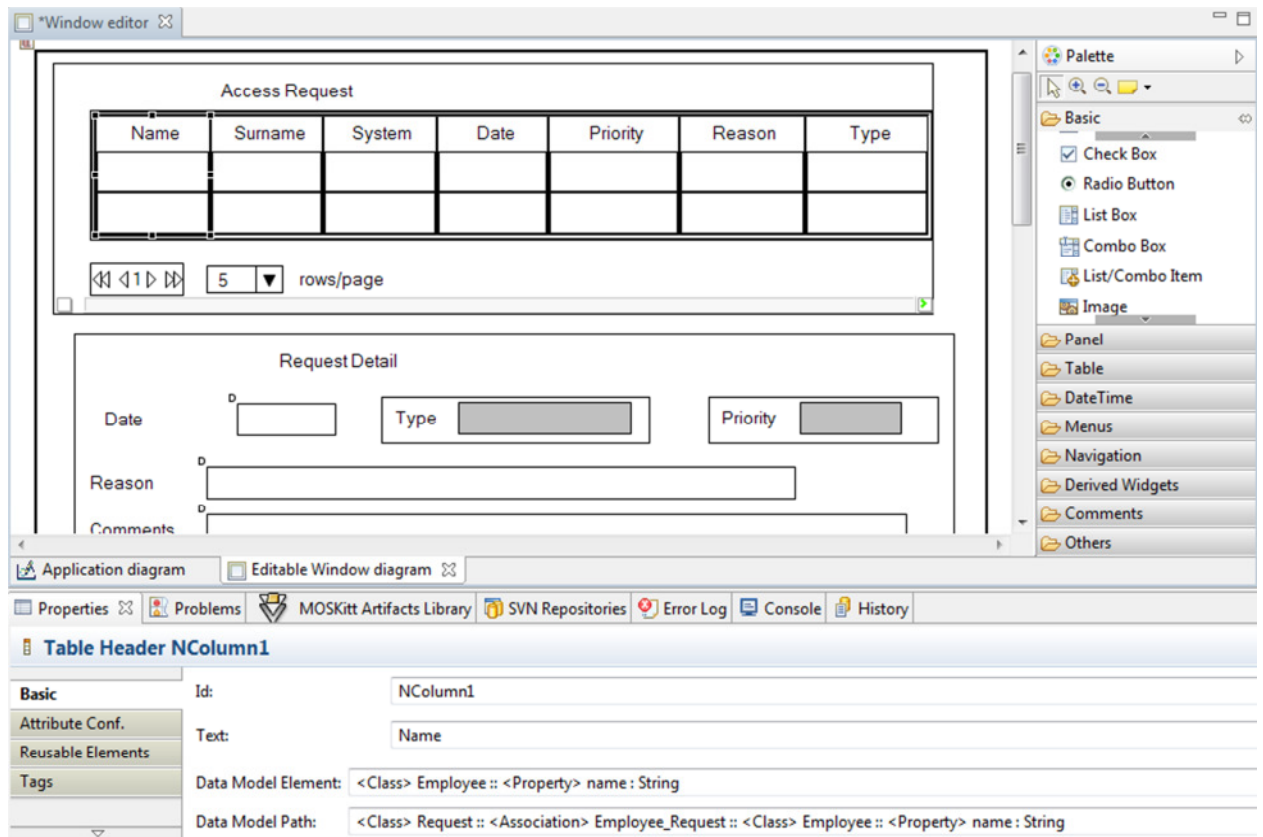


Figure 96: Sketcher diagram for the Request Module.

In Figure 97 is presented the generated web application in execution visualizing the list view of requests.

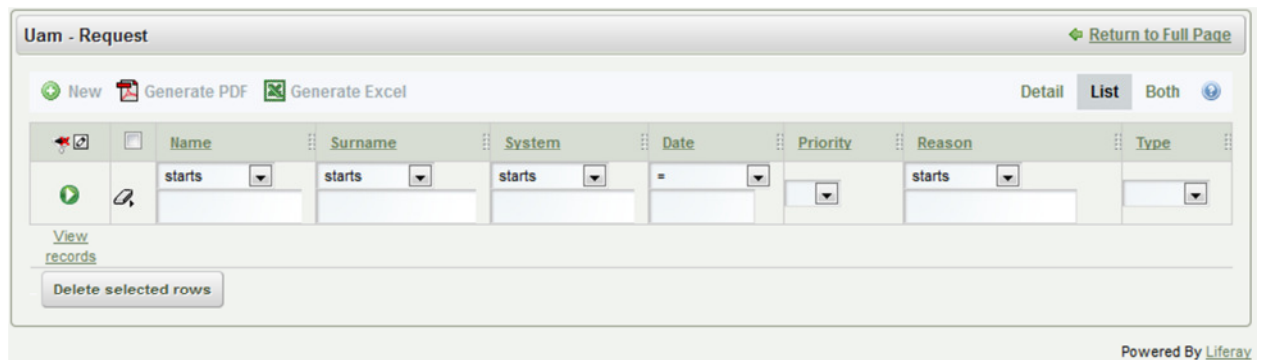


Figure 97: UAM application in execution, Request module.

## MOSKitt Transformation Details

### UML to Java transformations (UML2JPA)

Let us consider a simple example in order to explain the UML2JPA MOSKitt transformation. Figure 98 below shows part of the UAM model, the relationship between a “company” and its main “address”, modeled as a one-to-many association named *Company\_Address* and that represents the fact that a company has one main address [44]. Class *Company* contains three attributes: *name*, *website*, *nip* and class *Address* contains attributes: *street*, *zipCode*, *city* and *country*.

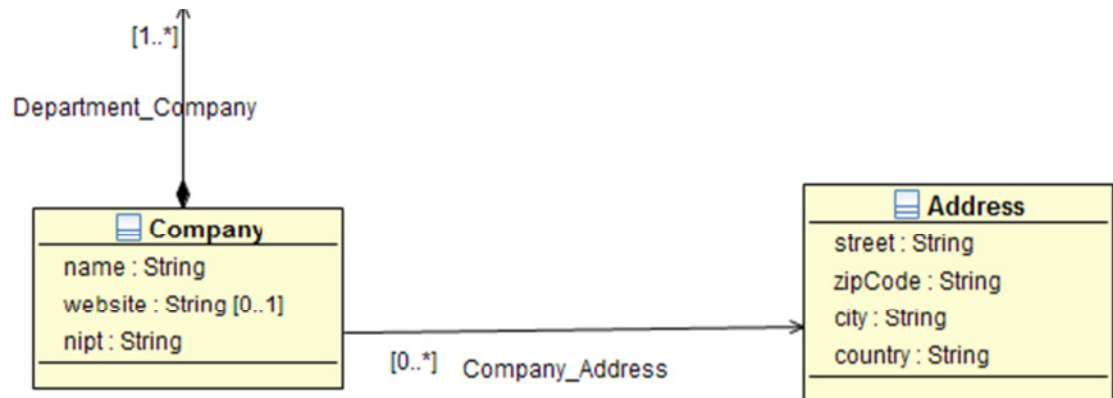


Figure 98: Company – Address relationship, UAM model.

Through the automatic transformation process an UML class will be transformed into a Java class. As an example **Company** class in the UML model in Figure 98 will be transformed into Java code as presented in Figure 99, representing the *Company.java* class.

The transformation rules are:

- A UML class will be transformed into a Java class.

- A class attribute in the UML model will be transformed into a private attribute in the respective Java class.
- For each attribute in the UML class two public methods (a get and a set method) are generated in the corresponding Java class.
- For each association end, there is a private attribute of the same name in the opposite class of type equal to the type of the class in the case when the association is *one to one*. In the event the association is of type *one to many*, then the type of the private attribute is *Set*. In the case the association is navigable (directed association) then the private attribute will be added only to the class origin of the association (in our example, class *Company*) also with the respective annotations (@ManyToOne).

During the transformation process the principle of encapsulation is respected; values of the attribute are accessed only by the objects itself. Therefore, the attributes of both classes *Company* and *Address* are declared private and the corresponding *get* and *set* methods are defined [33].

```
@Entity
public class Company {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long idCompany;

    @Column
    @NotNull
    private String name;

    @Column
    private String website;

    @Column
    @NotNull
    private String nipt;
```

```

@NotNull
@ManyToOne(cascade = {CascadeType.REFRESH}, fetch = FetchType.LAZY,
optional = false)
private Address address1;

@OneToMany(cascade = {CascadeType.REMOVE}, fetch = FetchType.LAZY,
mappedBy = "company1")
private Set<Department> department1;

/*
 * Gets the current value of the idCompany property
 */
public Long getIdCompany() {

    return this.idCompany;
}

/*
 * Sets the given value of the idCompany property
 * @param Long p0
 */
public void setIdCompany(Long p0) {

    this.idCompany = p0;
}
...

```

Figure 99: Company.java class obtained automatically through the UML2JPA transformations.

OpenXava is a business component framework because it allows defining all information about a business concept in a single place. A business component is a Java class. For example, for defining the concept of “company”, in OpenXava a single file *Company.java* is used, and all information about the company concept (including data structure, user interface layout, mapping with database, validations, calculations, etc.) is defined there [47]. The *Company.java* class obtained through the automatic UML2JPA transformation from the UML **Company** class, it is a business component, containing all the necessary information for the business concept.



Let's clarify how the data structure and database mapping is expressed in the generated Java class. The `@Entity` annotation indicates that this class is a Java Persistence Api (JPA) entity, in other words that the instances of this class will be persistent objects. A table with the name of the class will be created in the database and class attributes will be transformed in table columns of the same type (e.g. *A String will be mapped onto a SQL VARCHAR. A Date will be mapped onto a SQL DATE etc* ). In the *Company.java* class is generated also an attribute named *idCompany*. The respective annotations `@Id @GeneratedValue(strategy = GenerationType.AUTO)` indicate that this attribute is generated automatically and is the primary key of the respective table in the database. The other attributes of the object are created as per the specifications of the UML model. The attributes *name*, *website* and *npt* are created with *String* type and also `@Column` annotation indicating that these will be columns in the respective database table. There are present also two attributes generated automatically in order to model the associations of the **Company** class in the UML model. The attribute *address1* of type *Address* with annotations `@NotNull @ManyToOne(...)` represents the many-to-one directed association *Company\_Address* while the attribute *department1* of type *set<Department>* and annotations `@NotNull @ManyToOne(...)` represents the *Department\_Company* aggregation. The generated code in the Java class ensures communication with the database and data persistence. The database schema is generated automatically; the user has only to define the name of the database and some other simple parameters [33].

The execution of the resulting business component, *Company.java* in the developing environment allows to automatically generating a default user interface [33]. In case the

generated user interface is not satisfactory a specific user interface can be created through the Sketcher model.

## UIM2OX Transformations

Let's consider the Sketcher model presented in Figure 96 that defines the user interface for the *Request* module. The first transformation Sketcher2UIM is an intermediate model to text transformation of the user interface specified in the Sketcher diagram, allows obtaining a textual representation of the Sketcher Model, called a UIM model. The UIM2OX transformation allows generating the final Java code from an UIM model and JavaGenJPA model (a model composed by Java classes with JPA annotations; the generated so far Java classes) that when executed will display the specified user interface. Please refer to Figure 100 for the final version of the *Request.java* class updated with annotations necessary for the specific user interface.

```
package org.model.domain;

import java.lang.Long;

@Entity
@Tabs({

    @Tab(name =
        "Request.Access_Request_Tabular_View.Access_Request_Tabular_View.RequestL
        ist", properties = "employee1.name, employee1.surname, system1.name,
        date, priority, reason, type")
    )
    @Views({

        @View(name =
            "Request.Request_Detail_Registry_View.Request_Detail_Registry_View.Reques
            tDetail", members = "date, type; priority; reason; comments;
            accessPeriod1.startDate, accessPeriod1.endDate; Employee Details {
            employee1.name, employee1.surname, employee1.title; employee1.email,
            employee1.celPhone; employee1.department1.name;
            employee1.department1.profile; employee1.department1.company1.name,
```

```

employee1.department1.company1.website,
employee1.department1.company1.nipt;
employee1.department1.company1.address1.street,
employee1.department1.company1.address1.zipcode;
employee1.department1.company1.address1.city,
employee1.department1.company1.address1.country; } System {
system1.name, system1.description; system1.administrator,
system1.businessOwner; system1.roleGroup1.role1.name,
system1.roleGroup1.role1.description; system1.roleGroup1.identifier,
system1.roleGroup1.level; system1.roleGroup1.description; } ; ")}

)
public class Request {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long idRequest;

    @Column
    @NotNull
    private String reason;

    @Column
    @NotNull
    @Temporal(TemporalType.DATE)
    private Date date;

    @Column
    @NotNull
    @Enumerated(EnumType.STRING)
    private RequestType type;

    @Column
    @NotNull
    @Enumerated(EnumType.STRING)
    private RequestPriority priority;

    @Column
    @NotNull
    private String comments;

    @NotNull
    @ManyToOne(cascade = {CascadeType.REFRESH}, fetch = FetchType.LAZY,
optional = false)
    private Employee employee1;

    @NotNull
    @ManyToOne(cascade = {CascadeType.REFRESH}, fetch = FetchType.LAZY,
optional = false)
    private System system1;

    @NotNull
    @ManyToOne(cascade = {CascadeType.REFRESH}, fetch = FetchType.LAZY,
optional = false)

```

```

private AccessPeriod accessPeriod1;

@NotNull
@OneToOne(cascade = {CascadeType.REFRESH}, fetch = FetchType.LAZY,
optional = false)
private ApprovalPath approvalPath1;

/*
 * Gets the current value of the idRequest property
 */
public Long getIdRequest() {

    return this.idRequest;
}

/*
 * Sets the given value of the idRequest property
 * @param Long p0
 */
public void setIdRequest(Long p0) {

    this.idRequest = p0;
}
...

```

Figure 100: Request.java class with annotations for the user interface specified through the Sketcher model.

Note that the annotations `@Tab`, `@Tabs`, `@View`, `@Views` etc added to the *Request.java* class which are responsible to define the specific user interface as per the Sketcher model presented in Figure 96. Specific user interface design specified in the Sketcher Model is transformed in OpenXava annotations in the resulting .java class which can be interpreted and generate the desired user interface layout. The remaining of the .java class is the same as the one generated during the UML2JPA transformation.

### 9.3.1 Limitations

Starting from an UML Model (UML class diagram) through MOSKitt code generation it is possible to generate automatically a Java web application with CRUD (Create, Read,

Update, Delete) and Export functionalities without the need to write any Java code. The web application generated does not contain information regarding the specific business logic of the domain under study. In order to complete the OpenXava application generated automatically from the UML Model, incorporating into it business logic details, it's the responsibility of the application developer to write the necessary OpenXava code specifications. That is the reason why MOSKitt code generation is referred to as semi-automatic generator of OpenXava applications. It's a semi-automatic generation because the OpenXava code generated must be completed by the developers to include specific business logic for each specific application [34].

OpenXava framework does not include security, user management and module navigation. In order to add these features to an application though, there are several ways to do it [48]:

- **Integrate the application in a Java Portal such as Liferay, WebSphere Portal or JetSpeed:** OpenXava generates portlet applications that can be deployed in any standard Java Portal, so the application can be deployed in a Java portal and use the security, user management and navigation provided by the portal [51].
- **Use NaviOX:** NaviOX is an add-on for OpenXava that adds navigation, security and user management to applications easily [39].
- **Write the security and navigation code:** An OpenXava application is a regular Java web application, therefore security can be added in the same way as for any other Java web application.

- **Add spring security:** spring-security is an authentication and authorisation framework from spring framework [55]. No change in the OpenXava code is required.

## 9.4 Simulation in Bizagi Process Modeler

Bizagi Process Modeler provides the possibility to simulate business process as presented in section 7.2.4. In this section we take the User Access Management process model and simulate it through the first three simulation steps in order to understand possible improvements needed in the process and identify possible bottlenecks.

### Process Validation

The first level of simulation validates the Process Model, making sure the process passes through all the sequence flows, and behaves as expected. In the Process Validation level only Start Events and Gateways are enabled for editing. For this level we define for the User Access Management process the below properties based on figures from the company [9].

- **Max.arrival count:** Define the number of token instances the process will generate (or trigger). In our case we input 1000 as shown in Figure 101.
- **Gateways routing:** Inclusive and exclusive Gateways have activation probabilities. Probabilities are values between 0 and 100%. We define the

probabilities for the two exclusive gateways present in the user access management process with the values as shown in the figures 102 and 103 [9]:

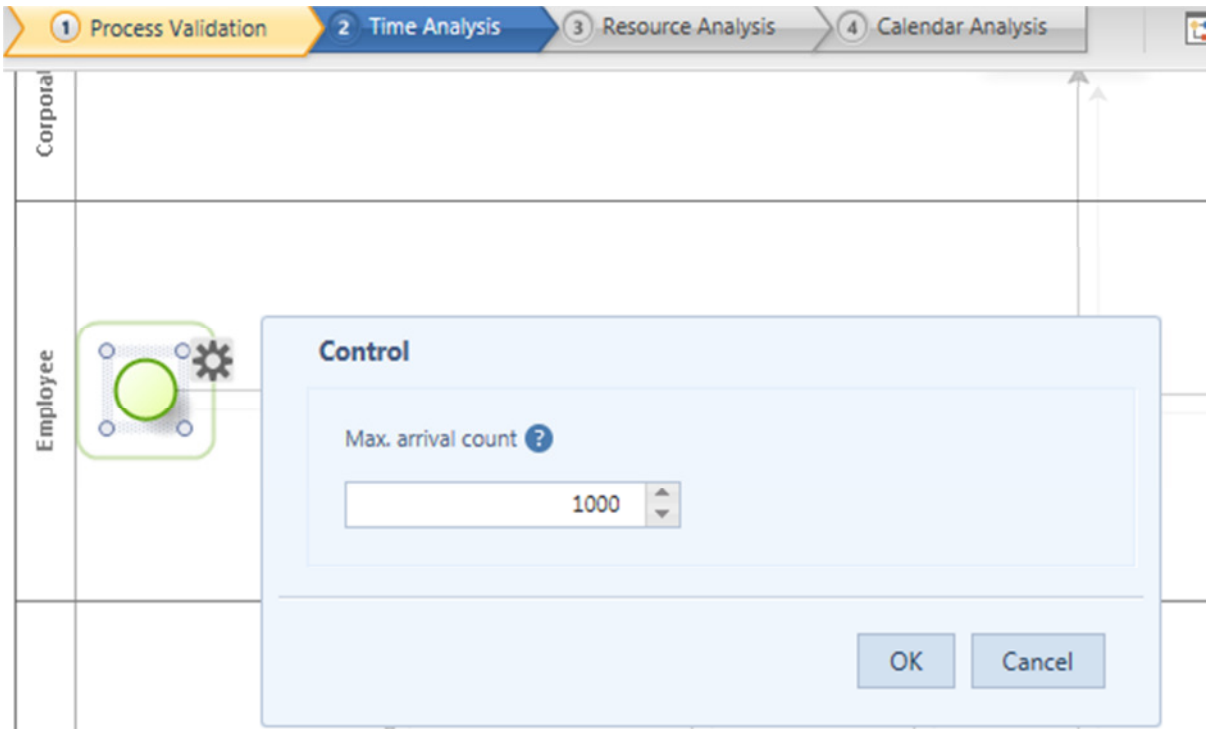


Figure 101: Max. arrival count for the user access management process.



Figure 102: Probabilities definition for the first exclusive gateway.



Figure 103: Probabilities definition for the second exclusive gateway.

We can run the simulation and verify that the process is validated. It behaves as expected and passes through all the sequence flows.

### Throughput time analysis

The second level of the simulation is useful in measuring end-to-end process times. Here, resources are not included. Bizagi assumes an infinite capacity to avoid delays in the process flow. This is the best case scenario under the given flow and processing times . In addition to the information specified in the previous level, the following must be defined in the throughput Time Analysis [9]:

- **Arrival interval time:** Defines the time interval between token instances generation. Instances will be created until the max.arrival count is reached. This applies to Start Events in our case and we define the time interval to 180 mins.



- **Processing times:** Defines the amount of time an Activity or Event needs to process a token. That is, it defines a service time period from the moment a token arrives at an Activity or Event until it is executed

Below we have defined the processing times for the activities in the user access management process based on figures extracted from the performance of the process and experience.

<b>Activity Name</b>	<b>Processing times</b>
<i>Enter Permission Request</i>	15 mins
<i>Approve PermissionsLM</i>	10 mins
<i>Approve PermissionsCM</i>	60 mins
<i>Inform Reject Reason</i>	1 mins
<i>Implement the request</i>	20 mins
<i>Inform Request Completion</i>	20 mins

### **Resource Analysis**

This analysis shows the potential effect of resource constraints on process performance. Remember that a Resource is defined as a person, equipment, or space necessary for the execution of a specific task. In the previous level, Time Analysis, we assumed infinite resource capacity, that is, activities are able to process infinite quantity of tokens at the same time. However this assumption is not practical at all. In real terms there

are always resources constraints. The most common issue arising from introducing resources constraints is that tokens need to wait to be processed at a given moment. By default the Performers defined in the process documentation are defined as resources [9].

In the Resources Analysis level we need to define the following parameters:

- **Resources:** The total resources available for the process. Remember that a resource is a person, equipment or space necessary for the execution of a specific task. The resources available for the user access management process are presented in Figure 104. We will not consider costs for the user access management process.
- **Resources requirements:** Tasks require resources to be performed. Once you have defined the process' resources, you have to define how many are required in order to perform a task.

Task Name	Resource	Number of Resources
Enter Permission Request	Employee	1
Approve PermissionsLM	Line Manager	1
Approve PermissionsCM	Corporate Security Manager	1
Inform Reject Reason	Employee	1
Implement the Request	IT Support Specialist	1
Inform Request Completion	Corporate Security Analyst	1

Resources	Quantities
Employee	400
Corporate Security Manager	1
It Support Specialist	20
Corporate Security Analyst	1
Line Manager	80

Figure 104: Resources defined for the User Access Management process

With all the information available so far we can run the simulation and analyze the results that are presented in Figure 105 in order to understand if there are any bottlenecks in the current operations of the user access management process.

Resources		Scenario information		
User Access Management		Name	Scenario 1	
		Time unit	Minutes	
		Duration	030,00:00:00	
Resource	Utilization	Total fixed cost	Total unit cost	Total cost
Employee	0.02 %	142,000	0	142,000
Corporate Security Manager	31.39 %	678,000	0	678,000
It Support Specialist	0.46 %	197,000	0	197,000
Corporate Security Analyst	9.12 %	197,000	0	197,000
Line Manager	0.07 %	480,000	0	480,000
	Total	1,604,000	0	1,604,000

Figure 105: Simulation results for the user access management process

From the analysis of the results it is clear that with the current workload on the process there are not bottlenecks. Considering the process flow that every request needs to be approved from the Corporate Security Manager we would suspect that this would be a bottleneck for the process. The results of the simulation show that the Corporate Security manager is not yet saturated and is not a bottleneck for the user access management process. The utilization of the Corporate Security Manager is 31.39%.

## 9.5 Comparison and conclusions

### Model Driven Development implementation

Starting from an UML Model (UML class diagram) through MOSKitt code generation it is possible to generate automatically a Java web application with *CRUD* (*Create, Read, Update, Delete*) and *Export* functionalities without the need to write any Java code. The web application generated does not contain information regarding the specific business

logic of the domain under study. In order to complete the OpenXava application generated, incorporating into it business logic details, it's the responsibility of the application developer to write the necessary OpenXava code specifications. That is the reason why MOSKitt code generation is referred to as semi-automatic generator of OpenXava applications. It's a semi-automatic generation because the OpenXava code generated must be completed by the developers to include specific business logic for each specific application [31]. *There is not a full compliance with MDA principles.*

In Bizagi, starting from a BPMN model and following the process automation wizard is possible to generate a web application that automates the business process without the need to write any code. *Bizagi business process management suite fully supports the MDA principles.*

### **Security, User Management and Module Navigation**

OpenXava framework does not include security, user management and module navigation. In order to add these features to an application though, there are several ways to do it as shown in section 7.3.1: Security, user management and module navigation are fully supported in Bizagi engine. The resulting application is executed without the need for further integrations or customizations.

### **Application Development**

Bizagi is a suite for process automation and is not applicable for general purpose application development. MOSKitt is a tool for the development of web applications,

suitable for business applications-oriented databases and not only process oriented applications.

### **Proprietary**

Bizagi is a proprietary solution. Processes automated with Bizagi can be executed only within the Bizagi Engine and respective license fees need to be paid for enterprise deployments. MOSKitt is a free platform using the EPL (Eclipse Public Licence).

### **Simulation**

Bizagi process modeler provides the possibility to simulate business models. Simulation is a very powerful mechanism for process improvement and optimization. MOSKitt platform does not provide such a capability

## **CHAPTER X: BUSINESS PROCESS MANAGEMENT PRACTICES IN THE TELECOMMUNICATION SECTOR IN ALBANIA**

### **10.1 Overview of the telecommunication sector in Albania**

The telecommunications sector in Albania is dominated from seven major operators that have an across the country presence. These operators are Vodafone Albania sh.a., Albanian Mobile Communications sh.a, Albtelecom sh.a., Plus Communication sh.a., Abcom sh.p.k., Abissnet and Albanian Satellite Communications (ASC). As per the latest report published from the National Authority of Electronic and Postal Communications there are around 100 operators present and operating in Albania. Only the above seven mentioned operators have an across the country presence while the other operators are very small and have only local or regional presence [40]. At the end of 2014 the sector of electronic communications in Albania has seen a decline in the number of subscribers of mobile telephony in active subscribers and in subscribers of active sim cards, respectively 3.359.654 subscribers with a year on year decline of -8.9% compared with 2013 and 4.881.666 subscribers with a year on year decline of -7.6% compared with 2013. Active subscribers refers to subscribers of SIM cards that have used mobile telephony services in the last three months while the number of subscribers as per SIM active cards refers to the number of SIM cards that are active in the mobile network at the end of the reporting period. A decline behavior is also present in fixed telephony with a year on year decline of -12.1%. The penetration of mobile telephony and fixed telephony is 123% and 9.2% of the

population respectively. On the other hand the fixed broadband market is in rapid growth with a year on year growth of 13.3% [40].

## Mobile Network Operators

In Albania operate four mobile network operators, Vodafone Albania sh.a., Albanian Mobile Communications sh.a, Albtelecom sh.a. and Plus Communication sh.a. and the penetration of mobile telephony in Albania is 123% of the population at the end of 2014.

The operators offer 3G services (Plus sh.a. is the exception) and are currently in the process of implementing 4G services across the country. In the below table is presented the market share for the operators as per active subscribers and subscribers of active sim cards [40].

Active Subscribers and Active SIM Cards	2010		2011		2012		2013		2014	
	Active	SIM	Active	SIM	Active	SIM	Active	SIM	Active	SIM
Plus	28,162	28,162	182,391	396,722	202,426	302,310	221,086	307,652	261,314	345,350
Albtelecom	602,775	822,356	415,307	1,110,363	435,922	1,423,421	424,237	676,679	431,772	675,901
Vodafone	1,129,715	1,674,748	1,358,871	1,809,264	1,491,126	2,019,655	1,659,697	2,239,612	1,549,498	1,966,101
AMC	1,396,752	2,022,541	1,152,518	1,920,000	1,407,811	1,874,221	1,380,963	2,058,407	1,117,070	1,894,314

Table 4: Number of active subscribers and active SIM cards per mobile operator from 2010 to 2014

In the below table is also presented the number of subscribers using 3G mobile broadband per operator.

Number of subscribers with 3G broadband	AMC		Vodafone		Albtelecom		Total	
	Cellphone	USB/Modem	Cellphone	USB/Modem	Cellphone	USB/Modem	Cellphone	USB/Modem
2012	222,882	17,833	371,426	37,572	NA	NA	594,308	55,405
2013	279,428	32,138	568,207	66,086	272,257	13,143	1,119,892	111,367
2014	163,797	61,549	503,855	46,341	197,220	15,170	864,872	123,060
YoY change 2014/2013	-41%	92%	-11%	-30%	-28%	15%	-23%	10%

Table 5: Number of subscribers with 3G broadband access per operator



## Fixed Telephony Operators

In Albania operate many fixed telephony operators due to the presence of many small local providers. The market is dominated from Albtelecom sh.a. (incumbent with 74% market share) and the other operators that have a certain relevance are Abcom sh.p.k, Albanian Satellite Communications, Albanian Mobile Communications sh.a. and Nisatel (local operator). In the below Figure 101 and Table 1 is provided a detailed picture of the sector. The penetration of fixed telephony in Albania is 9.2% of the population (around 31% penetration in terms of households) and the total number of subscribers are 247,070 at the end of 2014 as per the latest report with a decline of -12 year on year. The penetration of fixed telephony and its usage is in constant decline in Albania in line with the European trends on fixed telephony [40].

Number of Fixed Telephony subscribers per operator	Albtelecom	Abcom	ASC	Nisatel	AMC Fiks	Other Operators	Total
<b>2010</b>	277,763	19,975	7,408	5,900	n/a	20,456	331,502
<b>2011</b>	258,943	27,167	10,129	4,950	7,565	30,090	338,844
<b>2012</b>	230,397	33,000	7,649	5,500	6,119	28,996	311,661
<b>2013</b>	210,382	13,680	15,047	7,166	6,950	27,975	281,200
<b>2014</b>	182,591	15,531	15,047	9,236	3,106	21,559	247,070
<b>Year on Year change</b>	<b>-13%</b>	<b>14%</b>	<b>0%</b>	<b>29%</b>	<b>-55%</b>	<b>-23%</b>	<b>-12%</b>

Table 6: Number of Fixed Telephony subscribers per operator

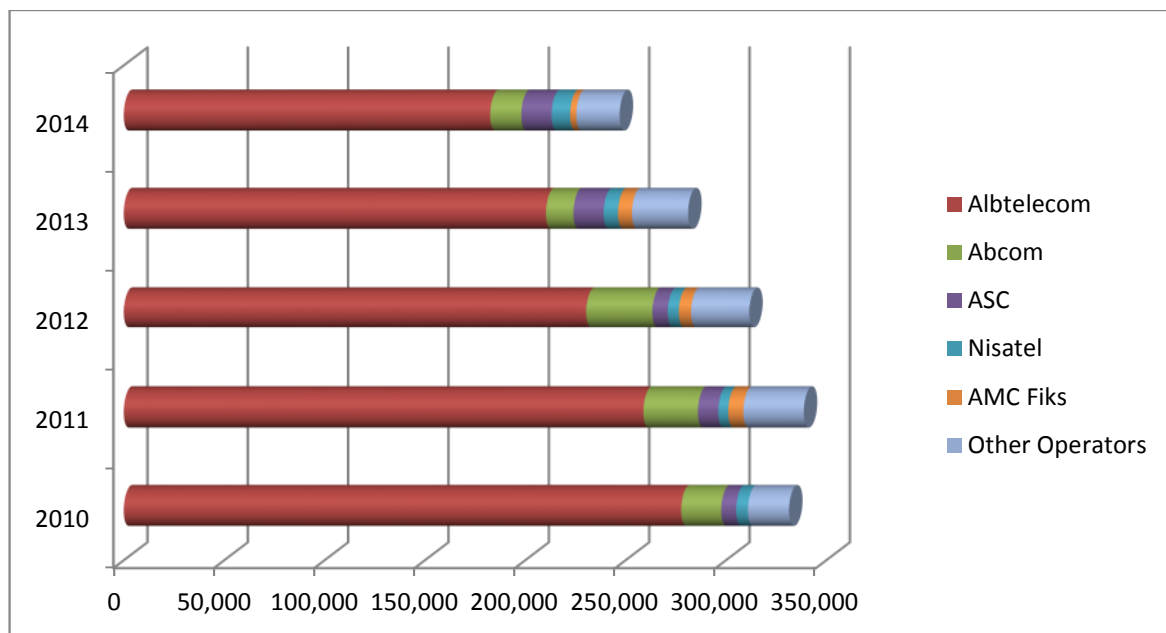


Figure 106: Number of Fixed Telephony subscribers per operator, end of 2014.

### Fixed Broadband Operators

The number of Fixed Broadband operators in Albania is also elevated due the presence of many small local providers. The market leaders in the fixed broadband market in Albania are Albtelcom sh.a. (incumbent), Abcom sh.p.k, Albanian Satellite Communications and Abissnet. The penetration of fixed broadband in Albania is around 27% of households and the total number of subscribers is 206.896 at the end of 2014 as per the latest report with a 13% increase year on year. The fixed broadband market in Albania has not reached its maturity and saturation and is in continuous expansion. In the below table is presented the market share for the operators at the end of 2014 [40].

<b>Number of subscribers with access to fixed broadband</b>	<b>Albtelecom</b>	<b>Abcom</b>	<b>Abissnet</b>	<b>ASC</b>	<b>Other operators</b>	<b>Total</b>
<b>2010</b>	70,597	13,575	7,000	5,666	23,162	120,000
<b>2011</b>	60,055	29,321	15,321	10,129	24,871	139,697
<b>2012</b>	66,757	35,870	17,719	11,777	27,965	160,088
<b>2013</b>	73,242	43,430	20,562	15,432	29,890	182,556
<b>2014</b>	82,118	47,480	23,259	26,379	27,660	206,896
<b>YoY change 2014/2013</b>	<b>12%</b>	<b>9%</b>	<b>13%</b>	<b>71%</b>	<b>-7%</b>	<b>13%</b>

Table 7: Number of Fixed Broadband subscribers per operator

Following the global trends on the telecommunications industry and the changes that are happening across the industry where operators are moving towards the offering of converged propositions (bundling several fixed services like fixed telephony, fixed broadband, Pay TV and mobile services together) also in Albania there is a tendency on offering converged propositions. All major fixed broadband operators have implemented and are offering Pay Tv services (IPTV or Cable TV) and also offer fixed bundled services (Fixed Telephony, Fixed Broadband and Pay TV). In particular Albtelecom is offering quad play services by bundling together Mobile telephony, Fixed Telephony, Fixed Broadband and IPTV in a single package.

## **10.2 Business Process Management practices**

For the scope of the study on business process management practices in the telecommunications market in Albania we will take into consideration only the seven major

across the country companies (Vodafone Albania sh.a., Albanian Mobile Communications sh.a, Albtelecom sh.a., Plus Communication sh.a., Abcom sh.p.k., Abissnet and Albanian Satellite Communications) since the rest of the operators are very small and with not a clear and defined company and operational structure. We are confident that such small local operators that are mostly family run businesses have not business process management practices in place. A survey was prepared for completion to responsible persons within the seven major telecommunication companies in Albania. Also in some cases to explain possible inquiries and understand in more details the situation interviews were performed. In the below section we will highlight the results of the survey for each area.

### **Business Process Modelling and Business Process Management awareness**

In all the major telecommunication companies in Albania there is a clear understanding and awareness for Business Process Management. The importance of Business Process management is recognized. In six out of the seven companies (85% of the cases) there is also an understanding regarding Business Process Modelling and its importance even though most of the times there is a clear association between Business Process Modelling and Business Process Management.

### **Responsible team within the company for Business Process Management**

In six out of the seven companies (85% of the cases) the companies have in places teams dedicated for Business Process Management with a clear structure and reporting to top management. This is clear evidence that top level executives in such companies recognize the importance of Business Process Management and Business Process Improvement.

### **Tools and formalisms used for business process modelling, BPMN**

In all the companies there is not used and is not in place a tool dedicated for Business Process Modelling. Applications such as Microsoft Visio or similar are used in some cases to model business processes but there are not in place business process modelling tools that support the BPMN standard or UML. Even though four of the seven companies state in the survey that they are aware of BPMN standard, in practice the BPMN standard is not used to model business processes.

### **Tools used for business process management, Bizagi**

In all the companies there are not in place dedicated Business Process Management suites that support business process modelling and business process management. The companies use applications/solutions provided or present in the company for business process management and in some cases to automate process flows. In some cases CRM or ERP applications are used for the purpose as well. In all the companies there are not BPMS suites in place to manage the business process lifecycle and that allow process automation starting from the process model following the model driven development approach. The importance of historical reporting in order to improve business processes is clear and understood in all cases but the historical capability of the applications it's not the same as the one provided from a BPMS suite. None of the companies uses Bizagi BPMS suite for business process management.

### **Tools used to simulate business processes**

In all the companies there are not in place dedicated solutions used for business process simulation and business process simulation is not a practice used from the business process management teams. Process improvement is based only on historical reports and common practice careful review of the processes.

### **10.3 Conclusions**

Following the results of the survey on business process management in the telecommunication sector in Albania it can be concluded that:

- There are no BPMS solutions used for business process management in the companies. The benefits and advantages provided from such tools are not entirely explored and we believe that the companies by investing in a business process management solution like e.g. Bizagi BPMS suite will have a clear advantage towards the competitors to streamline and better improve business processes.
- There is not a clear formalism used to model business processes and there are no tools that support it. The BPMN standard for business process modelling is not used to model business processes. We strongly suggest the adoption of the BPMN standard within the companies. We also recommend that such adoption should happen quickly within the companies and it would provide to the ones that adopt it a clear advantage towards the competitors. The utilization of the BPMN standard is a first step towards the progress of the companies in using a BPM suite. This first step

can be performed easily and with no costs since there are several tools that support BPMN to model business processes and are for free like Bizagi Process Modeler etc.

- Business process improvement is based mainly in historical reports and common understanding of the stakeholders involved in the process. Process simulation is not used to improve and optimize business processes. We strongly recommend the usage of process simulation as a must in order to proper optimize business processes. Process simulation can provide savings for the companies and at the same time guarantee the quality of the processes. The investment needed in such case from the companies is practically zero since there are several tools that support business process modelling and are free. One of them is Bizagi Process Modeler.
- Is important also to point out that all the companies are aware of the importance of business process management and have dedicated teams and functions in place that are responsible for business process management.

## REFERENCES

- [1] Bizagi BPM suite documentation, available at <http://help.bizagi.com/bpmsuite/en/>
- [2] Bizagi BPM suite documentation, Process modelling available at <http://help.bizagi.com/bpmsuite/en/>
- [3] Bizagi BPM suite documentation, Data modelling available at <http://help.bizagi.com/bpmsuite/en/>
- [4] Bizagi BPM suite documentation, Creating the user interface available at <http://help.bizagi.com/bpmsuite/en/>
- [5] Bizagi BPM suite documentation, Defining business rules available at <http://help.bizagi.com/bpmsuite/en/>
- [6] Bizagi BPM suite documentation, Work allocation available at <http://help.bizagi.com/bpmsuite/en/>
- [7] Bizagi BPM suite documentation, Security definition available at <http://help.bizagi.com/bpmsuite/en/>
- [8] Bizagi BPM suite documentation, Bizagi Engine available at <http://help.bizagi.com/bpmsuite/en/>
- [9] Bizagi Process Modeler documentation, available at <http://help.bizagi.com/processmodeler/en/>
- [10] Bizagi Process Modeler documentation, Modeling the process, available at <http://help.bizagi.com/processmodeler/en/>
- [11] Bizagi Process Modeler documentation, Documenting the process, available at <http://help.bizagi.com/processmodeler/en/>
- [12] Bizagi Process Modeler documentation Process Simulation, available at <http://help.bizagi.com/processmodeler/en/>



- [13] Bizagi Process Modeler documentation Simulation levels, available at <http://help.bizagi.com/processmodeler/en/>
- [14] Bizagi Process Modeler documentation Simulation scenarios, available at <http://help.bizagi.com/processmodeler/en/>
- [15] BPMN introduction by Bizagi, available at <http://www.bizagi.com/eng/downloads/BPMNbyExample.pdf>
- [16] BPMN v2. Documentation Guide, available at <http://www.omg.org/spec/BPMN/2.0/>
- [17] Business Process Management, available at [http://en.wikipedia.org/wiki/Business\\_process\\_management](http://en.wikipedia.org/wiki/Business_process_management)
- [18] Business Process Simulations standard, available at <http://www.bpsim.org/>
- [19] Clark T, Sammut P, Williams J: Applied Metamodelling: A foundation for language driven development, second edition 2008
- [20] Douglas P, Alliger G, Goldberg R (1996) Client-server and object-oriented training, 1996.
- [21] Eclipse platform, available at <https://eclipse.org/eclipse/>
- [22] Forrester Consulting, Modernizing Software Development through Model-Driven Development. A commissioned study conducted by Forrester Consulting on behalf of Unisys, August 13, 2008.
- [23] Frankel, S. D.: Model Driven Architecture. Applying MDA to Enterprise Computing. Wiley Publishing, Inc. OMG Press 2003.
- [24] Gerth C.: Business Process Models. Change Management, 2013 – Springer
- [25] Guttman M, Parodi J: Real-Life MDA: Solving Business Problems with Model Driven Architecture, (The MK/OMG Press), 2007.
- [26] Guttman M, Parodi J: Real-Life MDA: Solving Business Problems with Model Driven Architecture, (The MK/OMG Press), 2007.
- [27] Java annotations, available <http://docs.oracle.com/javase/1.5.0/docs/guide/language/annotations.html>

- [28] Jeston J, Nelis J: Business Process Management: Practical guidelines to successful implementation. Butterworth-Heinemann 2006.
- [29] Lusk S, Staci P, Spanyi A: The evolution of business process management as a professional discipline. June 2005.
- [30] Model Driven Architecture executive overview, available at [http://www.omg.org/mda/executive\\_overview.htm](http://www.omg.org/mda/executive_overview.htm)
- [31] MOSKitt code generation, available [http://www.moskitt.org/eng/openxava\\_que\\_es/](http://www.moskitt.org/eng/openxava_que_es/)
- [32] MOSKitt code generation installation guide, [http://www.moskitt.org/eng/openxava\\_instalation00/](http://www.moskitt.org/eng/openxava_instalation00/)
- [33] MOSKitt code generation, “Tutorial 3: How to launch an OpenXava application with OXPortal”, available [http://www.moskitt.org/eng/openxava\\_manual0/](http://www.moskitt.org/eng/openxava_manual0/)
- [34] MOSKitt code generation, “Tutorial 1: Developing an OpenXava application from UML2 MOSKitt models”, available [http://www.moskitt.org/eng/openxava\\_manual0/](http://www.moskitt.org/eng/openxava_manual0/)
- [35] MOSKitt code generation, “Tutorial 2: Including User Interface design in OpenXava applications with MOSKitt”, available [http://www.moskitt.org/eng/openxava\\_manual0/](http://www.moskitt.org/eng/openxava_manual0/)
- [36] MOSKitt Functional Modules, available <http://www.moskitt.org/eng/moskitt-modulos-funcionales/>
- [37] MOSKitt home page, available <http://www.moskitt.org>
- [38] MOSKitt User Guide Manual Version 0.7.0, available <http://www.moskitt.org/eng/manuales/>
- [39] NaviOX, available <http://www.openxava.org/naviox>
- [40] National Authority on Electronic and Postal Communication 4<sup>th</sup> quarter of 2014 report
- [41] Object Management Group (OMG), available <http://www.omg.org>
- [42] OMG Model Driven Architecture, available <http://www.omg.org/mda>
- [43] Object Management Group, Model Driven Architecture (MDA): The MDA Guide Rev. 2.0, 2014 available at <http://www.omg.org/mda/presentations.htm>
- [44] OMG Unified Modeling Language Infrastructure, available <http://www.omg.org/spec/UML/2.4.1/>

- [45] OMG, “MDA Guide Version 1.0.1”, available <http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf> 12th, June 2003.
- [46] OpenXava home page, available <http://www.openxava.org/>
- [47] OpenXava Reference Guide, available at [http://openxava.wikispaces.com/reference\\_en](http://openxava.wikispaces.com/reference_en)
- [48] OpenXava; Security, user management and module navigation, available at [http://openxava.wikispaces.com/security\\_en](http://openxava.wikispaces.com/security_en)
- [49] OpenXava, Wikipedia introduction, available at <http://en.wikipedia.org/wiki/OpenXava>
- [50] Owen M, Raj J: BPMN and Business Process Management: Introduction to the new business process modelling standard.
- [51] Paniza J.: Learn OpenXava by example, 2011.
- [52] Papajorgji P., Pardalos, P.: Towards a Model-Centric Approach for Developing Enterprise Information Systems.
- [53] Pastor O., Molina, J.C.: Model-Driven Architecture in Practice A Software Production Environment Based on Conceptual Modeling, Springer 2007.
- [54] Plain old java object definition (POJO), available at <http://www.martinfowler.com/bliki/POJO.html>
- [55] Spring security, available <http://static.springsource.org/spring-security/site/>
- [56] Unified Modelling Language standard, available at <http://www.uml.org/>
- [57] White S.A: Process Modeling Notations and Workflow Patterns, IBM Corporation 2006.
- [58] White S.A.: Introduction to BPMN, July 2014.