USING AN ENTERPRISE MODEL AS A REQUIREMENTS MODEL IN PROCESS AUTOMATION SYSTEMS: A PROPOSAL

Fábio Levy Siqueira (Escola Politécnica da USP, SP, Brazil)
levy.siqueira@usp.br

Paulo Sergio Muniz Silva (Escola Politécnica da USP, SP, Brazil)
paulo.muniz@poli.usp.br

Abstract: There are several options to represent computer systems requirements, such as use cases, requirements lists, user stories, goal diagrams, and formal specifications. Each requirement representation has advantages and disadvantages. Using the WRSPM model as a requirement framework, this paper separates the requirements model from the specifications model and proposes using an enterprise model as a requirements model. Based on a specific type of system – process automation systems – a meta-model for an enterprise model with an organizational perspective is proposed, and an example of its practical use is also presented. This work is contextualized in a proposal to transform requirements into specifications using Model-Driven Engineering concepts.

Keywords: Enterprise model, process automation systems, requirement, requirements engineering, WRSPM model.

Acknowledgement: This work was partly supported by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), grant no. 2007/58489-4.

1. Introduction

Nowadays, computer systems are essential for enterprises to participate in an increasingly competitive market. Computer systems not only help employees and other stakeholders to execute different tasks, but they can also automate these tasks. Although computer systems are very important, the enterprises themselves consist of much more. There are several other elements involved, both physical and non-physical, such as employees, tools, tasks, artifacts, goals, and missions. As computer systems are just one of these elements, they must be created based on the context within which enterprises will operate. If not, the expectations that motivated the construction of the computer systems in the first place may go unfulfilled – and, as a result, the systems may fall into disuse.

In Requirements Engineering, information about the elements within an enterprise is analyzed differently through various approaches, with the aim of understanding what clients want to obtain from the system. Some approaches consider this information indirectly, for example, by means of interviews, requirements workshops, brainstorming, and storyboarding. In these techniques the stakeholders naturally consider several elements within the enterprise that are related to the system, even if they are not explicit. In other approaches only

selected types and/or a summary of enterprise elements are considered in order to elicit requirements, for instance, in i* (YU, 1995), Tropos (BRESCIANI et al., 2004), and KAOS (VAN LAMSWEERDE, 2009). On the other hand, some approaches specifically use enterprise models for requirements elicitation, as in EKD (BUBENKO JR.; BRASH; STIRNA, 1998).

In general, requirements obtained during elicitation are not expressed in any useful manner that can be employed by other software development activities. Some of them must be detailed to be guaranteed or executed by a computer (ZAVE; JACKSON, 1997), which can involve a selection from among alternatives. The choice between any such alternatives depends on the context within which the system will participate, which involves the context of the enterprise.

In its search for a requirements model that can adequately represent the context of the enterprise and can be used for automatic transformation to detailed requirements – called here specifications –, this paper proposes an enterprise meta-model to be used as a requirements model. This meta-model has an organizational perspective of the enterprise and was obtained by analyzing several studies in the area of Administration. The hypothesis considered here is that, for some types of systems, this more detailed view of the enterprise helps to define more adequate requirements and enables them to be transformed into specifications.

To present this meta-model, this paper is organized as follows: in section 2, the requirements framework used by this work, the WRSPM model (GUNTER et al., 2000), is presented. Next, in section 3, the relationship between the WRSPM model and an enterprise is discussed, and the use of an enterprise model as a requirements model is proposed. In section 4 a meta-model based on the analysis of several organizational representations is proposed. This section also presents the tool created to describe models in conformance to the proposed meta-model, and discusses the research goal to be achieved with this meta-model. In section 5, related works are presented; and in section 6, a book library model that uses the proposed meta-model is discussed. Finally, in section 7, our conclusions and proposals for future works are presented.

2. Requirements Reference Model

There are several terms that represent ideas related to requirements and differentiate important information in the Requirements Engineering discipline. Therefore, terms such as function, features, need, domain knowledge, and several others are defined, along with definitions of requirement. Among the existing definitions, this work will use the ideas presented by Jackson and Zave (JACKSON, 1995) (JACKSON, 1997) (ZAVE; JACKSON, 1997), using the WRSPM model (GUNTER et al., 2000), presented in Figure 1, as a reference. The goal of this model is to provide a common basis for discussing software development artifacts, and their attributes and relationships. Therefore, the artifacts are classified into five groups considering their importance to the environment – the part of the world with which the system will interact, and where the effects of the system will be observed and evaluated (JACKSON, 1997) – or to the system itself. These artifacts are (GUNTER et al., 2000):

- domain knowledge, with known and presumed information about the environment;
- requirement, indicating what the client needs from the system, described as effects on the environment;
- specification that enable the system that satisfies requirements to be built;
- the program that implements the specification; and
- the platform that enables the system to be programmed.

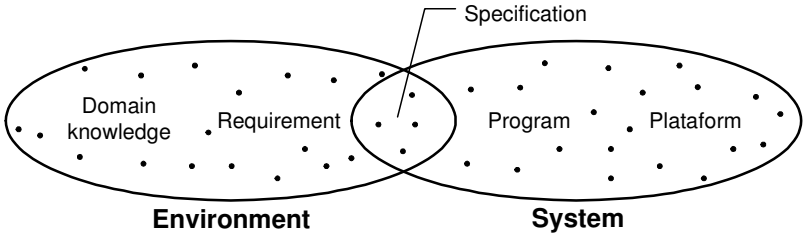

Figure 1. The WRSPM model (GUNTER et al., 2000).

Each of these artifacts can be viewed as a description that uses a specific vocabulary comprising of primitive terms. Some of those terms are shared between artifacts, as the phenomena that they aim to denote are also shared. These phenomena are the dots in Figure 1 and they represent states, events, and individuals which are organized into four classes by the model: the ones that belong to the environment and are visible or invisible to the system, and, similarly, the phenomena that belong to the system and are visible or invisible to the environment. Following this division, only the visible phenomena are shared between the environment and the system. For example, in a library computer system, an environmental phenomenon that is invisible to the system is the arrival of new books; a visible environmental phenomenon is the registration of these new books; a system phenomenon visible to the environment is the availability of these books for lending; and, finally, an invisible system phenomenon is any change to the database record regarding these books.

Given these classes, the domain knowledge artifact restricts the environmental phenomena. In addition, it can also restrict the relationship between environmental phenomena and phenomena that belong to the system that are visible to the environment (GUNTER et al., 2000); for instance, in the example of the library system, the restriction is that a book becomes unavailable when it is lent. The requirement artifact adds other restrictions to the ones defined by the domain knowledge artifact, considering client necessities. In other words, the requirements are decided by the clients. Using the library example once more, a restriction such as a library member only being allowed to borrow three books at any given time can be part of the requirement artifact. On the other hand, the platform artifact restricts the phenomena that can be executed by the system, while the "program" artifact describes a class of possible system phenomena with regards to the program. In the middle of the environment and the system there is the "specification" artifact that uses their common vocabulary to determine what the system must accomplish with regards to the environment.

For the sake of simplicity, each artifact in the WRSPM model is considered here as comprising of a set of statements, and those statements are called domain knowledge for the domain knowledge artifact, requirement for the requirement

artifact, and specification for the specification artifact. Consequently, "requirement" here is defined as a statement that describes "the environment as we would like it to be and as we hope it will be when the machine is connected to it" (ZAVE; JACKSON, 1997, p.7). On the other hand, specification is a specific type of requirement that only considers the phenomena shared by the environment and the system (JACKSON, 1995). It is a type of requirement refined by removing all the characteristics that prevent its implementation. Therefore, the specification is a requirement that does not need any additional information in order to be used by the development team in their subsequent tasks (such as analysis, design, code, etc.).

The specification is obtained from the requirements through a process called refinement (ZAVE; JACKSON, 1997). This process consists of two steps: "(1) identifying the aspects of a requirement that cannot be guaranteed or effected by a computer alone and (2) augmenting or replacing them until they are fully implementable" (ZAVE; JACKSON, 1997, p.19). Regarding the first step, there are three general reasons for requirements not being considered specifications, according to Zave and Jackson (1997): they constrain phenomena controlled by the environment instead of constraining phenomena controlled by the system; they are described in terms of unshared phenomena (not visible to the system); and they are described in terms of the future. In all these cases, in order to transform requirements into specifications, it is necessary to use the domain knowledge, and to more precisely define the problem space – as well as the boundaries between the system and the environment. Nevertheless, there is generally more than one possible refinement, with several alternatives possible. Each alternative can impact differently on the proposed system, influencing desired business or technical results (MYLOPOULOS et al., 2001), or even the project (time, cost, or the size of the team needed) itself. They also restrict differently the solution space, with different options involving what the system should do. Therefore, such alternatives must be discussed with the stakeholders, and they should be evaluated considering pre-established desired results, and negotiated with the decision makers (VAN LAMSWEERDE, 2009).

3. Relationship between the Enterprise and the WRSPM Model

According to Vernadat (1996, p.22), an enterprise is "a socio-economic organization created to produce products or to procure services and to make profit". In the context of system development, an enterprise model is normally used to understand the structure and dynamics of the existing organization, to guarantee that the stakeholders have a common knowledge of the organization, or to understand how new systems can aid in increasing productivity and how the existing systems will be affected by it (LEFFINGWELL; WIDRIG, 2003). That is, the enterprise model is not normally used as a requirements model. Generally an environmental model is used for this purpose, as suggested by Zave and Jackson (1997).

Considering the scope of computer systems development, an enterprise is clearly part of the environment. The system will offer support or will be a means for the enterprise, or part of it, to achieve its ends, and will hence directly affect it. Therefore, the enterprise is one of the main stakeholders of the system, evaluating

the adequacy of the system to its business. Moreover, the system will interact with parts of the enterprise, such as its employees, and other systems that exist within it. Nevertheless, although the enterprise is part of the environment, it is not always the entire environment. There are usually other entities that are affected by the system and that evaluate it, often interacting with the system. For example, there are regulatory agencies (who define standards, laws etc.), partners or clients that will use the system, or even other systems with which the system will interact. Therefore, even if the system is custom developed for the enterprise, the environment usually includes other phenomena outside the enterprise itself.

On the other hand, the system can be seen as part of the enterprise. It will be one of the elements of the enterprise that will allow it to attain its established ends and to participate in its business processes – as with other systems that are inside it. However, this does not necessarily mean that a system must be restricted to only one enterprise. It is actually possible for a system to be spread over several enterprises, such as a travel scheduling system, where hotel reservations are made by one enterprise and airline tickets are bought by another. Depending on the viewpoint, it is possible to see this example differently: each subsystem can be seen as part of an enterprise (considering a management subsystem), or it can even be considered a single enterprise created by the business relationship of these two enterprises. Consequently, defining the boundary between the enterprise and the system under consideration is fundamental.

Based on the WRSPM model, the relationship between the environment, the system, the enterprise, and the requirements is presented schematically in Figure 2. The enterprise, presented as a dashed line, has phenomena that are shared with the environment and the system. This relationship can vary depending on the system. In the extremes, the enterprise boundary can include the whole system (when the system will be inside the enterprise), or the entire environment (when only the enterprise will be affected by the system). In other situations, the environment and the system can include the whole enterprise (when the entire enterprise will be affected by the system, although with other participants). But in general there are phenomena that are specific to the enterprise, in other words, that are not visible to the environment and neither to the system, as well as environmental and system phenomena that are invisible to the enterprise.

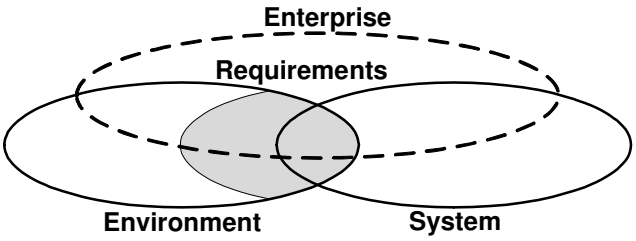

Figure 2: Enterprise, requirements and the WRSPM model.

As the requirements can be viewed as statements about the environment, some of these statements will also be statements about the enterprise. These common statements are about information or enterprise elements that are directly related to the system or that are affected by it. Nevertheless, those statements are not necessarily requirements: some of them may be domain knowledge. As some parts of the enterprise might not be affected by the system (not being part of the

environment), some statements about the enterprise may neither be domain knowledge nor requirements. Just as some statements about the enterprise are not necessarily requirements, some requirements are not necessarily statements about the enterprise. When part of the system is exclusively related to another enterprise, some requirements are naturally statements about it – and not about the represented enterprise. Usually the enterprise will have some requirements, and some of them will be outside the enterprise. Since specifications are a specific type of requirement in the boundary of the enterprise and the system, they may either be inside or outside the enterprise boundary.

3.1. Using an Enterprise Model as a Requirements Model

If some statements about the enterprise are statements about the environment, and these statements describe the environment as the stakeholders would like the system to be, or in other words, requirements, then an enterprise model – that is, a set of statements about an enterprise, using the definition of model presented by Seidewitz (2003) – can be seen as a requirements model. If the enterprise statements are about specifications then the enterprise model could be considered a specifications model.

Although the enterprise model can be seen as a requirements or a specifications model, it should only be used as one of these models if there are some advantages for Requirements Engineering activities and if any potential disadvantages do not make its use unfeasible. Perhaps the main advantage in using the enterprise model for this purpose is that it aids in system-business alignment. The introduction of a new system causes an impact on the enterprise, affecting the other elements involved in different ways. However, this impact is not always what was initially idealized or what may have motivated the development of the system, either because organizational goals were not adequately considered during Requirements Engineering activities (SIKDAR; DAS, 2009), or because the environment considered during the development of the system is different from the environment during deployment (THEVENET; SALINESI, 2007), or even because the professionals of the enterprise do not use the system as they should (RAMOS; BERRY; CARVALHO, 2005). Accordingly, one concern during system development is to obtain an alignment between the system and the business. One way of obtaining this alignment is by incorporating enterprise modeling into the Requirements Engineering activities (DECREUS; POELS, 2008) (SIKDAR; DAS, 2009), or, more broadly, by considering an adequate enterprise model during these activities (CHAMPION; MOORES, 1996). Therefore, using an enterprise model as a requirements or specifications model is one possible way of achieving this alignment, as the requirements would be related directly to the enterprise environment.

Besides the business-system alignment, using an enterprise model as a requirements or specifications model also enables the requirements to be contextualized, thus representing their origins within the enterprise. The possibility of relating requirements to the elements of the enterprise enables decisions to be traced (BUBENKO JR.; BRASH; STIRNA, 1998), and because it also represents the origins of requirements, it is also possible to understand any changes in enterprise needs (YU; MYLOPOULOS; LESPERANCE, 1996). Nevertheless, depending on what representation is used, this information can be highlighted to a

greater or lesser extent. For instance, desired results can be related to requirements – similarly to goal-oriented approaches –, requirements can be related to business processes in which the system participates, or requirements can be related to other enterprise information.

Finally, another advantage is that the enterprise model can facilitate communication between the requirements engineer and the domain specialists (DE LA VARA; SANCHÉZ; PASTOR, 2008). As the domain specialists usually do not have experience with software development, some representations are more difficult to understand and it is consequently more difficult to validate certain requirements (DE LA VARA; SANCHÉZ; PASTOR, 2008). In contrast, an enterprise model can be expressed in a language closer to the one used by these specialists, thus making it easier to achieve common understanding and avoid errors in the definition of requirements.

On the other hand, there are several issues or disadvantages in using enterprise model as a requirements or specifications model:

- Other stakeholders define requirements. As a result, only some of the requirements will be represented in an enterprise model. To have a complete requirements model, it is also necessary to consider the model of other participants.
- Difficulty in representing some requirements. The language used in an enterprise model might not have the adequate terms to represent requirements. Functional requirements will only be described if the language somehow enables the use of the system to be represented. Since some non-functional requirements do not have a direct effect on the environment, they will only be represented if the qualities they address are relevant to the language used by the enterprise model.
- Excess of information that is irrelevant to the system. Several model elements of the enterprise neither interact with the system nor are affected by it – i.e. they will not be part of the environment. In addition of being irrelevant, the excess of information can even hinder the use of the enterprise model for this purpose.
- Difficulty in creating and maintaining an enterprise model. It is not always simple to create an enterprise model, as it may consist of several elements and the relationship between them can be very complex.

Even though these issues make it difficult to use the enterprise model as a requirements or specifications model, some considerations or even some heuristics may alleviate the impact of these issues. The fact that the enterprise model is an incomplete requirements description can be overcome if it is considered that this model is just a partial description – like a use case model, for example. To use it, it is necessary to add some further information that enables the other requirements to be represented. The same argument can be used for the difficulty to represent some types of requirements. Regarding excess of information, one possible solution is to simplify the model by summarizing some information when using it during the Requirements Engineering activities. This solution would also facilitate the creation and maintenance of an enterprise model, thus addressing this other issue.

Nonetheless, there are other problems in using the enterprise model as a specifications model. Firstly, the enterprise meta-model would have to represent the details of the relationship between the system and the environment, which is necessary for a specification. A specific enterprise meta-model would be necessary to cover this information, which would mitigate several previously-discussed advantages. Second, as the specification is on the border between the environment and the system, it must be described more technically in order to be used by the development team. Therefore, it must be more direct, and not represent some of the information of the enterprise model that is not relevant to development. Therefore, even if the enterprise model can be used as a specification model, as specifications may be represented in an enterprise model, it does not seem to be suitable for this kind of use.

3.2. Process Automation Systems

To use the enterprise model as a requirements model, it must contain statements about the environment where the computer system is. Consequently, the representation of the enterprise should be suited to the type of system being developed.

Because of the relationship between the enterprise and the system, some important requirements may not be represented in the model, even if the representation is adequate. This situation occurs, for instance, in a distributed system in which a small part is managed and used by another enterprise. If an enterprise model is used as a requirements model, then several requirements will not be represented. Hence, the use of the enterprise model as a requirements model is limited to a class of problems. In general, enterprise modeling in Requirements Engineering is more useful when the application domain is complex and there are several people directly involved in using the system (LEFFINGWELL; WIDRIG, 2003). It seems to be the same when using it as a requirements or specification model. When the impact of the system is limited to a few elements of the enterprise, the enterprise model will only represent a small part of the requirements, and will not be that useful. Likewise, if algorithms and mathematical formulas are very important to the system, then the enterprise model will also not be a good requirements or specifications model.

Therefore, the scope of this work is a specific type of system, with the following characteristics:

- Automation of business processes, focused on workflow; and
- Inside an enterprise, but possibly with some interface with other participants.

For the purposes of this work, this type of system will be called *process automation system*. Since these systems emphasize information exchange with other elements of an enterprise and are inside the enterprise in question, the enterprise comprises the entire system and a large part of the environment (it can comprehend the entire environment when the system is not affected by external elements). The hypothesis considered here is that enterprise models can represent a majority of the process automation system requirements.

4. Enterprise Meta-Model

For an enterprise model be able to represent a majority of the requirements in the context of process automation systems, it is necessary to use a meta-model that is able to represent the elements of the enterprise that are relevant to the system. The first idea would be to use the business process as an enterprise model. However, it is just one view of an enterprise – although a very important one. If just the business process is considered, it may not be able to represent enough requirements in an enterprise model. Another possibility would be follow goal-oriented approaches and represent the enterprise through its motivations. Nonetheless, the same argument against the business process model could be used here, as this representation is limited to only a view of the enterprise. Although concepts such as plans and resources are also represented, they are simplified.

With the purpose of attaining a more comprehensive representation of the enterprise, in Administration, Organization and Methods (O&M) is traditionally responsible for the institutionalization of the infrastructure needed for an enterprise to achieve its goals, and for the definition and redefinition of work processes and methods (CURY, 2007). These responsibilities are accomplished using an enterprise model, either as a means of executing the activities of process and improvement of administrative methods, or as the result to be obtained (when an administrative manual is created).

Even though O&M represents a classical vision of Administration, the fact that it does not emphasize computational systems seems to be especially appropriate to deal with process automation systems. This kind of system usually substitutes work that is currently being done manually. Even if computational systems support employees, the intended objective of the new system is to absorb part of the manual activities. Hence, the existing computational systems will either be absorbed or used as part of an automated activity. In contrast to the O&M view, a more modern view, for example, enterprise architecture, emphasizes the relationship between business and Information Technology. Therefore, the enterprise models created within this area naturally consider details pertaining to the system – for example, the technological architecture view of TOGAF (THE OPEN GROUP, 2009). In contrast, the enterprise model used in the O&M context does not consider computational systems as a central element, having a broader view of the enterprise. As a result, their models represent documents, processes, hierarchical relationships, etc.

4.1. The Proposed Meta-model

Each O&M approach defines a different set of elements to be represented by an enterprise model. Although there is a variety of diagrams and elements, it is possible to notice some common diagrams and elements when analyzing different studies related to the area (ADDISON, 1971) (ANDERSON, 1980) (CRUZ, 2005) (CURY, 2007) (HARRINGTON, 1993) (LERNER, 1992) (OLIVEIRA, 1994):

- **Organization chart**: represents the enterprise hierarchy, describing the different levels of authority and responsibility of its elements (ADDISON, 1971) (HARRINGTON, 1991).

  - Content: position, department, function, task, relationship, and hierarchical relationship.
- **Flow chart**: represents the work or the document flow among the several workers (CURY, 2007) (LERNER, 1992) (OLIVEIRA, 1994).
  - Content: document, operation, check operation, transport, wait, file, flow direction, decision, and position.
- **Layout**: represents the distribution of furniture, equipment, people etc. in the organization space (CURY, 2007) (LERNER, 1992).
  - Content: layout and equipment.
- **Standards, policies, and directives**: in the studies considered, there is no agreement as to the meanings of these terms (mainly, the difference between them), but they usually correspond to established criteria that formalize the business of the enterprise, and they are organized and described in manuals (ANDERSON, 1980) (CURY, 2007) (LERNER, 1992) (OLIVEIRA, 1994).

Another element that is not usually represented at length in diagrams, but which is vital to an enterprise from an O&M standpoint, is the form (or document). In general O&M studies propose methods to create them, which involves diagramming, printing, selecting paper quality and color etc. (ADDISON, 1979) (ANDERSON, 1980) (CURY, 2007) (OLIVEIRA, 1994). Diagrams, such as the flow chart, describe its methods of use and storage, but there are not usually any diagrams or other representations that describe its internal details – perhaps because a form is a physical element that is easily manipulated. Despite this lack of a specific diagram, the information contained therein is vital to the enterprise and an enterprise representation must somehow contain it.

From these main elements it is possible to create an enterprise meta-model. According to Bézivin (2006, p.41), a meta-model "describes the various kinds of contained model elements, and the way they are arranged, related, and constrained."; that is, a meta-model is a model that defines the modeling language used by another model (KLEPPE; WARMER; BAST, 2003). There are some possibilities to represent a meta-model, for example, the ones presented in (ATKISON; KÜHNE, 2003) and (BÉZIVIN, 2006). In this work, a representation with concrete syntax, abstract syntax, and semantics used by OMG standards, like UML (OMG, 2010b) and BPDM (OMG, 2008), will be employed. Concrete syntax deals with the external representation of a language (MEYER, 1990), that is, its notation. Abstract syntax defines the concepts, relationships between them, and formation rules that define how they can be combined (CLARK; SAMMUT; WILLANS, 2008). Finally, semantics define the meaning of the language (MEYER, 1990). According to this representation, to define the enterprise meta-model abstract syntax, it is necessary to list the elements found and create some others to obtain a model that is consistent; with regards to concrete syntax, it is necessary to decide if it will have one or more diagrams and select the notation used; and with regards to semantics, it is necessary to find a common meaning for each element and adapt it, taking into account the other elements. An option to simplify meta-model creation is to use standards that deal with the group of elements or diagrams found. More than helping to define the syntax and semantics of a subset of the meta-model, using standards also helps to obtain something close to a consensus in a part of the enterprise meta-model. With

regards to the flow chart, there are several standards such as the IDEF3 (MAYER et al., 1995), UML activity diagram (OMG, 2010b), and Business Process Definition Metamodel (BPDM) (OMG, 2008). With regards to the standards, policies, and directives there is the Business Motivation Model (BMM) (OMG, 2010a) standard that provides a meta-model for developing, communicating, and managing business plans. However, there are no standards that could be used with regards to the other views[1].

Using this concept of absorbing standards, the meta-model will be defined using the BPDM and BMM meta-models. These standards were chosen because they are business oriented, and they are recent and provide a clear description of a meta-model (without being just a notation). With regards to the other views, the elements presented previously will be used, with the addition of further elements to make the meta-model consistent. Due to lack of space and in order to make the meta-model simpler to understand, only the abstract syntax will be represented, as a UML class diagram. For the same reasons, the OCL constrains needed to describe the full abstract syntax will not be represented.

In order to make understanding the defined elements easier, 5 views were defined, which are organized according to the elements and diagrams found: Organizational, Process, Layout, Motivation, and Document. Each of these views is organized into packages. When a certain element is referenced by a different view, it is represented by its qualified name.

4.1.1. Organizational View

In Figure 3, part of the abstract syntax for the organizational view is presented. *Entity* is the main meta-class that was created to represent the generalization of *Department* and *Position*. A *Position* is part of a *Department*, and works for it. *Entity* has links with other entities, represented by the *EntityRelationship* meta-class. To represent the hierarchy between the entities, there is a specific type of EntityRelationship, the *HierarchicalRelationship*. An *Entity* also has *Functions* that represent the attributions it has in the enterprise context. This *Function* is carried out by a *PerformerRole*, which is responsible for executing *Activities* in a *Process*. This meta-class is described in the Process view, in the BPDM meta-model.

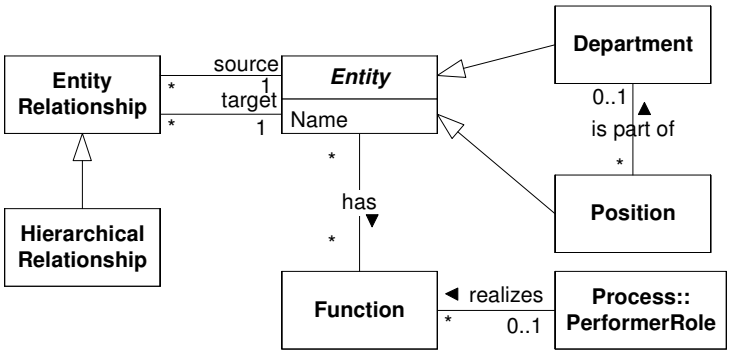

Figure 3. Organizational view.

[1] There is a standard being developed by OMG pertaining to a description similar to an organization chart: the Organization Structure Model (OSM). However, because there is no version of this standard available, it will not be used.

4.1.2. Process View

As the BPDM meta-model is used as a process view, in Figure 4 a fragment of this standard is presented. As the meta-model is very extensive, only some main meta-classes will be discussed herein. The meta-model is described in (OMG, 2008).

A *Process* specifies the *Activities* to be performed. An *Activity* activates a behavior when it is executed. Furthermore, it also interacts with other elements, with inputs and outputs, as it is an *InteractivePart*. An *Activity* that does not consist of by other *Activities* is a *SimpleActivity*. An *Activity* is the responsibility of a *PerformerRole* and it can be delegated to other *PerformerRoles*. The role represented by a *PerformerRole* is played by an *Actor* and there is also a higher level role, the *ProcessorRole*.

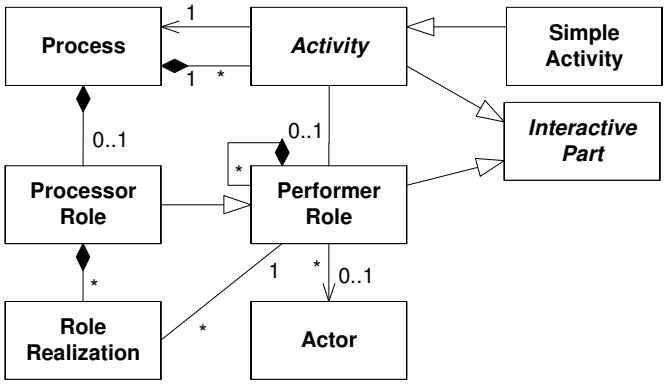

Figure 4. Fragment of BPDM (OMG, 2008), used as the process view.

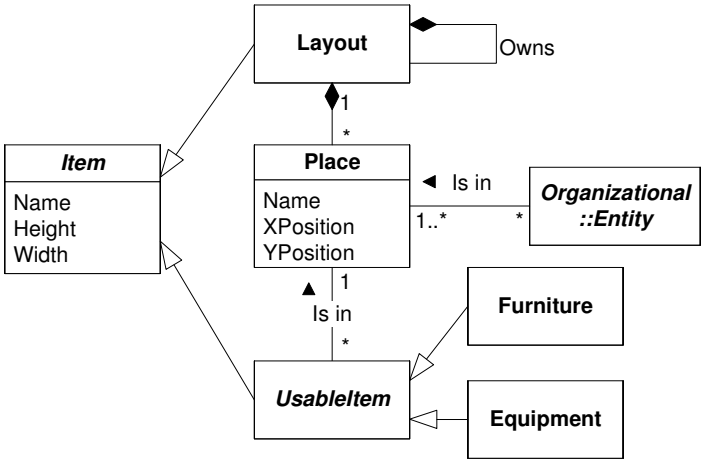

Figure 5. Layout view.

4.1.3. Layout View

In the Layout view, represented in Figure 5, an *Entity* of the organizational view is in one or more *Places*. Even though a *Place* is not a common concept in the studies analyzed, it is necessary to represent the physical location of an element – that is the goal of a layout diagram. Each *Place* has a *Name*, an X Position, and a Y Position in a specific *Layout*. A *Layout* can be owned by another *Layout* and so on, successively. *Equipment* represents tools and machines, and similarly, there is the *Furniture* meta-class that enables a complete office layout to be represented. Both *Equipment* and *Furniture* are *UsableItems* (an abstract meta-class), being in

a place and being able to be used by a *SimpleActivity*. Finally, there is the *Item* abstract meta-class that represents a *Layout* or a *UsableItem*. Each *Item* has a Name, a Height, and a Width.

4.1.4. Motivation View

In Figure 6 it is presented part of the motivation view, based on the Business Motivation Model (OMG, 2010a). The central element is the *MotivationElement*, with a name and a description. From this element, it is defined *Means* and *Ends*. *Ends* represent what the business aims to accomplish and the *Means* are what the enterprise does to achieve an *End* (OMG, 2010a). A type of *End* is the *DesiredResult*, representing a state or a target that the enterprise wants to maintain or sustain (OMG, 2010a). This *DesiredResult* can include other *DesiredResults* and can be specialized as a *Goal* and an *Objective*. The difference between these concepts is that an *Objective* is attainable, time-targeted, and measurable, while a *Goal* tends to be long term, and qualitative (OMG, 2010a). As the *Objective* is more precise, it quantifies *Goals*. *Means* can be specialized as *CourseOfAction* and *Directive*. The *CourseOfAction* is an approach or plan that channels efforts to a *DesiredResults*. It can include another *CoursesOfAction* and it also enables other *CoursesOfAction*. A *CourseOfAction* can be specialized as a *Tactic* and a *Strategy*. A *Strategy* is an essential *CourseOfAction* to achieve an *End*, while a *Tactic* implements the *Strategies*.

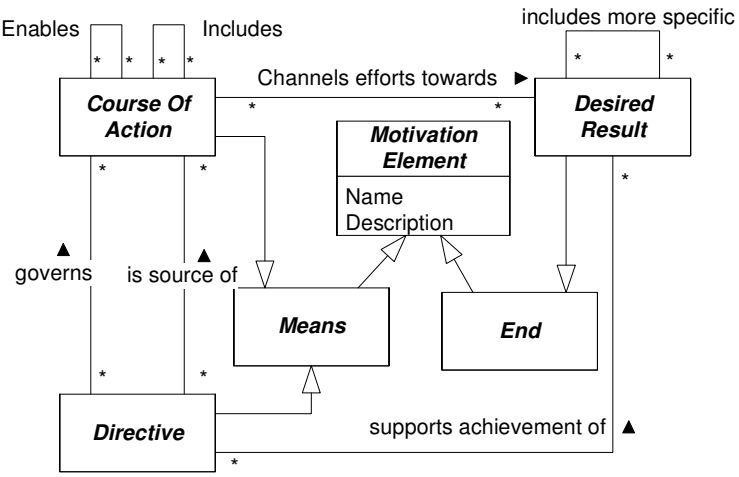

Figure 6. Part of BMM (OMG, 2010a), used as the motivation view.

As well as being a source of a *CourseOfAction*, *Directives* also indicate how it should be carried out and governed. Moreover, they also support the achievement of *DesiredResults*. The *Directive* meta-class is specialized in *BusinessPolicy* – that guides or governs an enterprise – and *BusinessRule* – that governs, guides, or influences business behavior and is directly applicable and atomic (OMG, 2010a). A *BusinessPolicy* includes more specific *BusinessPolicies* and are the basis of several *BusinessRules*. Finally, the application level of a *BusinessRule* is affected by the defined *Tactics*.

4.1.5. Document View

The main meta-class in the document view, described in Figure 7, is the *Artifact*. It has a name and several *ReservedSpaces* that must be filled in. Each *ReservedSpace* has a description. The *Artifact* is represented in a *Media* that can

be paper, CD, or other (described in the attribute *Type*). A specific *Artifact* is a *Form*, which has a *Field* as a *ReservedSpace*.

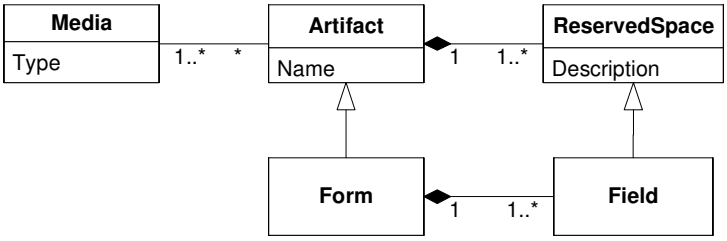

Figure 7. Document view.

4.1.6. Relationship between Views

The relationship between the five parts of the enterprise meta-model proposed here is described in Figure 8. Other relationships are also presented in Figure 3 and in Figure 5. Some of these relationships were obtained from BMM, as the meta-model defined has a relationship with other concepts. Thus, an *Entity* (from the organizational view) establishes *Means*, defines *Ends*, determines *Strategies*, and plays a *ProcessorRole*. A *Process* is realized by a *CourseOfAction*, then it is guided by a *BusinessRule* and governed by *BusinessPolicies*. Besides the BMM relationships, it was also considered that an *Entity* is in a *Place* and a *SimpleActivity* carries out a *Movement* of *Artifacts* from one *Place* to another (other types of movement were not defined to keep the model simple). A *SimpleActivity* can also use a *UsableItem*, and can read and write documents, accessing or filling in *ReservedSpaces*. Finally, a *ProcessorRole* can be played by *Software*, which represents the system boundary.

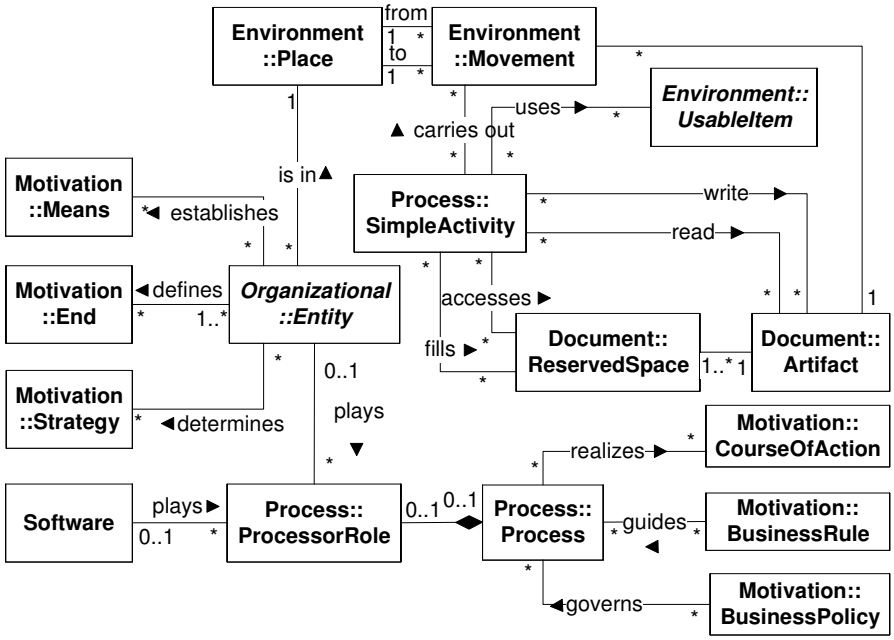

Figure 8. Relationship between views.

4.2. Tool Support

A tool was created to support the description of an enterprise model compliant to the proposed meta-model. A screenshot of this tool is shown in Figure 9.

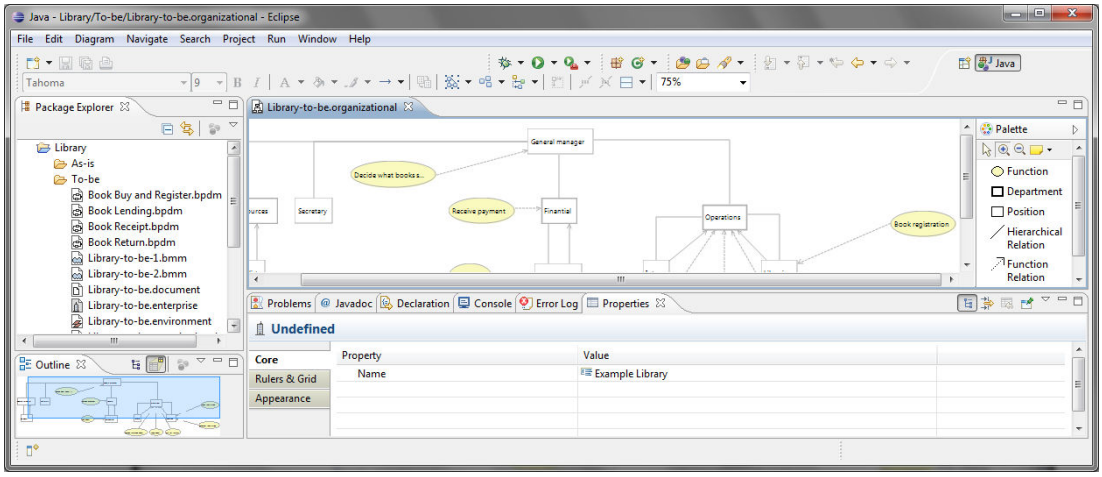

Figure 9. The interface of the tool created.

The tool was developed as a plug-in for the Eclipse IDE (THE ECLIPSE FOUNDATION, 2010), using the Eclipse Graphical Modeling Framework (GMF). This framework allows defining a concrete syntax from an abstract syntax, generating the infrastructure needed to create diagrams that describes a model coherent to the given abstract syntax. Therefore, we defined a concrete syntax for every view: the BPMN standard (OMG, 2011) was used for the process view, as it is compatible with BPDM; a notation based on goal approaches such as i* (YU, 1995) and KAOS (VAN LAMSWEERDE, 2009) was used for the motivation view; and a notation used by O&M organizational charts was used for the organization view. Specific notations were defined for the layout and the document views.

Using this tool it is possible to express several diagrams for every view, and use elements from one view in another (for instance, the same artifact is used in the document view and process view). The obtained enterprise model can be used in transformations to obtain other models of the system.

4.3. Transforming Requirements into Specifications

The enterprise meta-model described previously was created with the intention of transforming it automatically or semi automatically into specifications. The idea is to define rules that map elements of the enterprise model (meta-classes, meta-attributes, and/or relationships) to elements of a specification model (considering a use case meta-model), i.e., to apply a meta-model transformation (MILLER; MUKERJI, 2003). These rules are being written using the Operational Mappings language defined by the Query/View/Transformation specification (OMG, 2009). For instance, one of the proposed rules, described in natural language, is "A *PerformerRole* executing *SimpleActivities* that exchange (sends or receives) *MessageFlow* with a *SimpleActivity*, executed by a *ProcessorRole* carried out by the System, is an Actor of the use case meta-model." This rule was created based in the analysis of some studies, such as (DE LA VARA; SÁNCHEZ; PASTOR, 2008), (DIJKAMN; JOOSTEN, 2002), and (SANTANDER; CASTRO, 2002). Because some requirements must be refined in specifications, some rules are not only a change in the notation used. They absorb an appropriate refinement alternative, considering the context (process automation systems) and the domain knowledge available in the enterprise model. However, not all the information

defined by the enterprise meta-model will be useful for the transformation. Likewise, other information may be needed. It is an objective of this ongoing research to find out which elements are relevant, and which are not.

The transformation is based on the concept that the information needed for requirement refinement in specifications is available in the environment (specifically domain knowledge and requirements, as discussed in section 2). Therefore, the enterprise model will comprise of two parts: an as-is and a to-be model. The to-be model considers the system as one of its elements and, as discussed in section 3, it has some of the requirements and part of the domain knowledge. On the other hand, the as-is model only represents the domain knowledge, since this model does not includes the system in the enterprise environment. However, the domain knowledge described in the as-is model is different from the one presented in the to-be model: unless there is a complete reengineering of the enterprise, the attributions of the computational system in the to-be enterprise will be represented in the as-is model as attributions of other elements of the enterprise. Because this kind of domain knowledge may be useful for the rules, both the as-is and the to-be models are considered in the transformation.

There are some benefits of executing a requirements refinement as a transformation. First, it would facilitate the avoidance of mistakes during the refinement of requirements. Even a semi-automatic transformation would aid the requirements engineer task. Furthermore, a change in the requirements or in the environment, which is common in the context of an enterprise, would be more easily managed. If the change only impacts the requirements, it is necessary to revisit the refinement carried out; if the change only impacts the decisions taken during refinement, those decisions should be reviewed; and if the change impacts only the specifications, neither the requirements nor the refinement need be reviewed. Thus, it is important to represent both the requirements and the specifications, along with the decisions taken during the refinement, what is possible by using a transformation. If the requirements are not represented, the system may be inadequately restricted, and if the specifications are not represented, it may be difficult to execute subsequent development activities (MAIDEN, 2008). Finally, the definition of such a transformation would also improve from a theoretical standpoint the understanding of the relationship between requirements and specifications.

The details of this proposal and the rules for requirements to specification transformation will be discussed in future works.

5. Related Work

There are several proposals that use enterprise data to define requirements and/or specifications, but few of them actually use the enterprise model as a requirements model. For example, Champion and Moores (1996) propose an enterprise meta-model to be used for Requirements Engineering activities. That particular meta-model covers the process, organization, and motivation views of our meta-model. Nonetheless, that enterprise model is not a requirements model, although it is used to create one. Similarly, other proposals execute the

transformation from an as-is environment, pertaining to an organization, into requirements. For instance, in (DIETZ, 2003) and (DIJKMAN; JOOSTEN, 2002) an enterprise meta-model is used to obtain a requirements model represented as use cases.

In contrast, goal-oriented approaches, such as i* (YU, 1995), Tropos (BRESCIANI et al., 2004), and KAOS (VAN LAMSWEERDE, 2009), use a representation of the environment as part of its requirements model. However, there are some important differences between these approaches and the idea proposed in this work. Firstly, these approaches do not pertain to an enterprise, but an environment that is limited by the agents involved. Additionally, these approaches do not clearly separate the requirements from the specifications: the goal diagrams represent them both. This solution has the advantage of clearly representing the refinement, thus facilitating its execution (specially the choice between alternatives). Finally, these approaches mainly consider the motivation view of an organization, and have a simplistic description of the other's views. In contrast to these approaches, in the EKD (BUBENKO JR.; BRASH; STIRNA, 1998) approach, it is possible to clearly represent the relationships between the enterprise and the requirements (but not the specifications). Nevertheless, this approach does not use the enterprise model as a requirements model. The proposed meta-model has a sub-model regarding specifically to the requirements, and other sub-models regarding to enterprise concepts.

Some other works use an enterprise model as a requirements model. De La Vara, Sanchez, and Pastor (2008) propose an approach that uses a detailed representation of an enterprise. However, that representation is used to analyze organizational issues and needs. The result of that analysis is a labeled business process model – which is the one used as a requirements model. That model is then detailed into use cases that are used as specifications. As a result, the enterprise model used is limited to the process view alone.

An approach that is more similar to this work is proposed by Molina et al. (2000). They propose a method that uses an enterprise model to obtain use cases (used as specifications). That enterprise model is represented by business use cases, role diagrams, process diagrams, and a set of business rules. However, as the work does not have an organizational view of the enterprise, the meta-model proposed is very different from the one used in this work. Also, as the method proposed by Molina et al. (2000) emphasizes the business process, there are several representations that cover the process view. The organizational view is covered in more detail by the role diagram, but the agents are also represented in other diagrams. An important difference in the meta-model proposed by Molina et al. is that the business rules can represent concepts from the application domain, which cannot be represented by our enterprise model. Nevertheless, the motivation view is not covered by the meta-model, even if it is possible to define restrictions pertaining to enterprise-related goals and objectives.

Finally, several works discuss the transformation of parts of the enterprise model that was considered by this proposal. For example, Decreus, Snoeck, and Poels (2009) discuss approaches that transform a goal model into business process models, analyzing them from a methodological and organizational standpoint. According to the authors, this type of transformation is important because the goal

model enables the problem space to be represented, while the business process model can represent both the problem and the solution space, thus describing how the organizational goals are achieved. Nonetheless, in this work these models were considered to have complementary information about the requirements.

6. Example

The use of an enterprise model as a requirements model will be presented here by means of a simple and hypothetical example. Because the goal is to transform this model into specifications, as previously discussed, this model will comprise of an as-is and a to-be model.

The example that will be used here is a library that intends to automate book lending services and the management of returned books. Because of space limitations, this example will only be partially represented, thus limiting the representation of the enterprise. The library organizational structure is presented in Figure 10. This model represents both the as-is and the to-be model, because the organizational structure is not changed due to the system in the to-be model. The *Entities* are rectangles (a simple border represents a *Position* and thick border represents a *Department*), *HierarchicalRelationships* are links without direction, *Relationship* of pertinence are an arrow, and ellipses are the *Functions* of an *Entity*. The general manager is responsible for the library. An operational department is under the responsibility of the general manager, and in this department both the intern and the librarian can receive returned books, lend books, and ask for returns – but only the librarian can register a new member and new books.

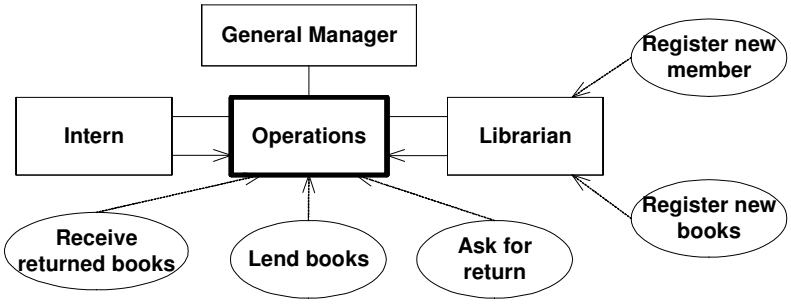

Figure 10. Library Organizational structure (as-is and to-be).

With regards to processes, only the "receive returned books" process is presented, with BPMN as the concrete syntax (OMG, 2011). The as-is process is present in Figure 11. In this process, an attendant (*PerformerRole* of an employee in the operations *Department*) gets the book, and obtains the checkout card and the member card. If the book is overdue, he calculates a fine and confirms receipt on the checkout card. He also informs whether the book is overdue on the member's card, at the same time, informing the member and forwarding the member to the accounts receivable department. If the book is not overdue, he confirms receipt on the checkout card and the member's card, which effectively confirms receipt.

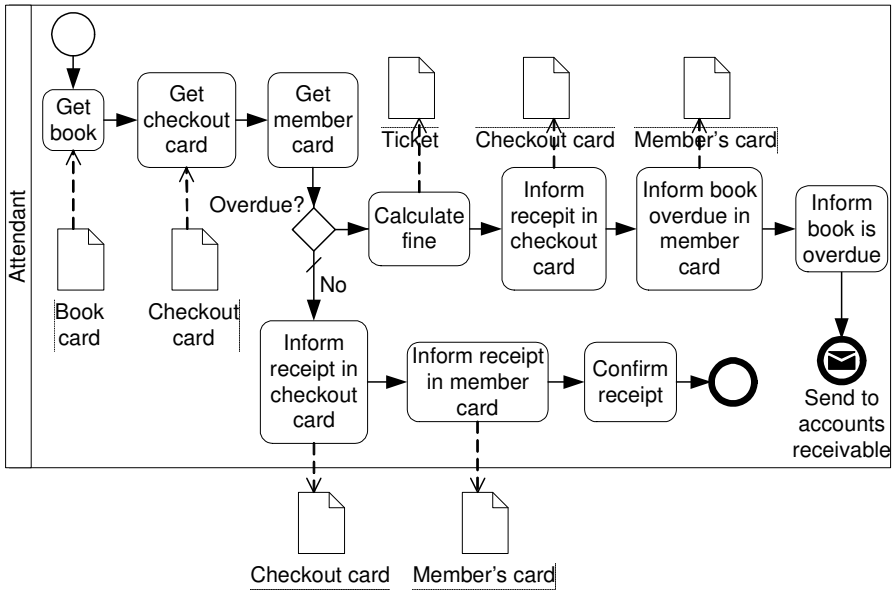

Figure 11. Library "receive returned books" as-is process.

In the to-be process, represented in Figure 12, the attendant gets the book and informs the system that it has been returned. The system analyses the return and informs the attendant. If the book is overdue, the attendant advises the member. If not, he only confirms receipt.

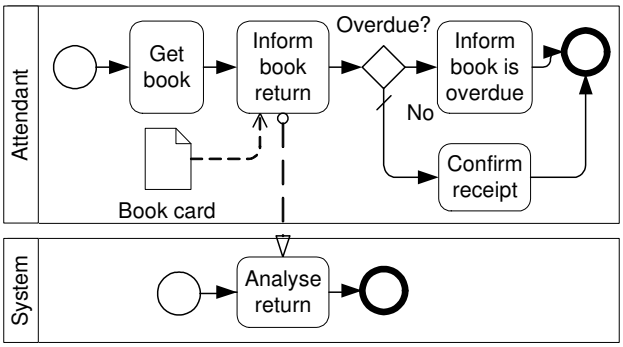

Figure 12. Library "receive returned books" to-be process.

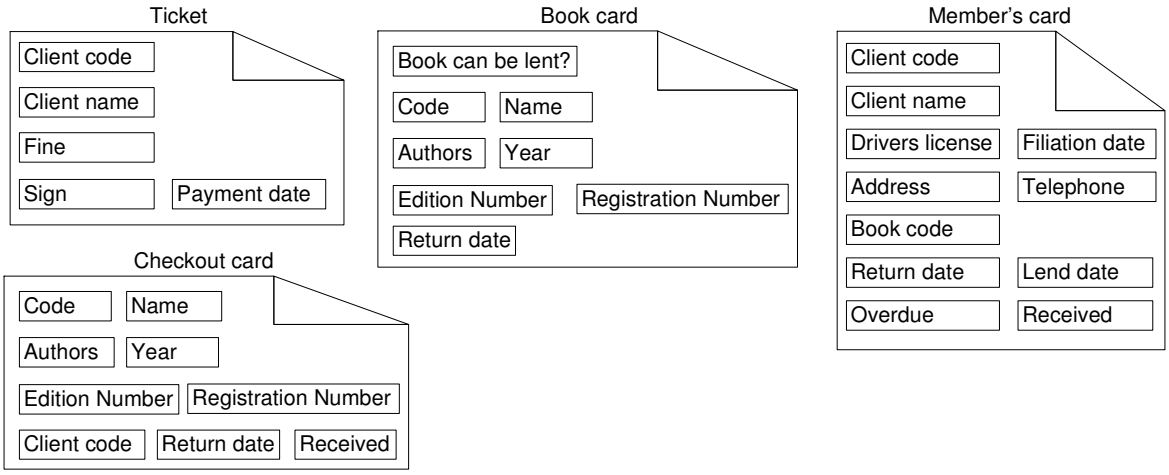

Figure 13. Part of the library as-is document view.

The document view is represented in Figure 13. It has the following *Forms* in the as-is model: book card, checkout card, member's card, and ticket. Each one of them has several fields. For instance, the book card has the following *Fields*: whether or not it can be lent, a code, the book title, the author(s), the year of publication, the edition number, a registration number, and Fields for return dates. In the to-be model there is only a book card, which contains the same Fields as in the as-is model.

In the layout view represented in Figure 14, the *Layout* is the same in the as-is and to-be models. It has a main *Layout*, representing the library, with five internal *Layouts*: the book area, the bathroom, the balcony, the table area, and the management office. Each *Layout* has several *UsableItems*, for instance, in the book area there are shelves and books.

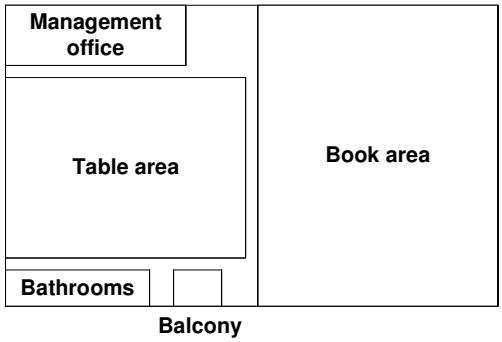

Figure 14. Part of the library as-is and to-be layout view.

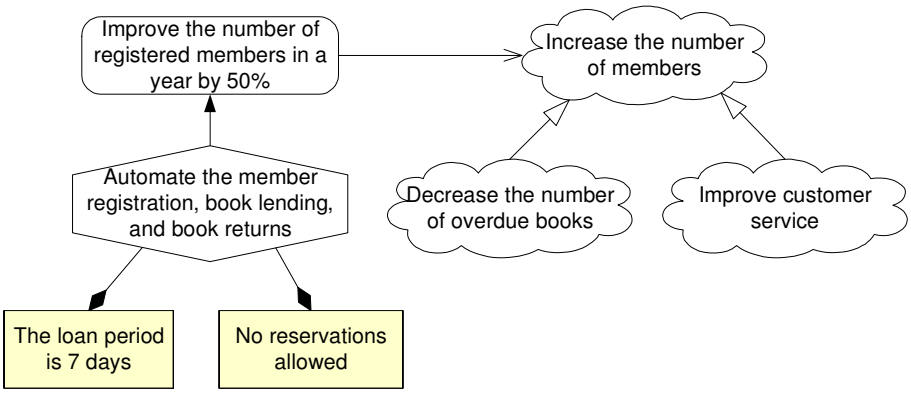

Figure 15. Part of the library as-is and to-be motivation view.

Finally, the motivation view contains several *Goals*, represented as clouds, such as "increase the number of members", which is represented in Figure 15. This *Goal* includes two more specific *Goals*: "decrease the number of overdue books" and "improve customer service". This *Goal* also is quantified by the *Objective* "improve the number of registered members in a year by 50%", represented as a round rectangle. The *Tactic* "Automate the member registration, book lending, and book returns" (represented as a hexagon) channels efforts towards this *Objective*. Also, there are some *BusinessRules* (represented as rectangles) related to this *Tactic*, for example, "no reservations allowed", and "the loan period is 7 days".

6.1. Discussion

Although only a small part of the enterprise model was represented, it is possible to observe several details regarding the requirements. With regards to the organizational model, in the example there were no changes between the as-is and to-be model. However, this is not always true. In another situation, a *Function* could have been removed from an *Entity*, for instance, "request return". In that situation, the function would be executed by the system – which would not be so evident in the to-be model. Nevertheless, it would be represented as a single activity process, executed only by the system. Another detail about the organizational model is that it represents some of the key stakeholders, with regards to the functions they will have in the system.

The processes seem to be a very important part of the requirements, as they clearly express how the enterprise must work – and, therefore, they highlight several functional requirements of the system. The *Artifacts*, from the document view, highlight some of the information needed for the process to be executed, which will also be important to the specification. With regards to the motivation view, the *BusinessRules* seem to represent important information for the requirement. For instance, the loan period is not expressed directly in the process diagram, but is defined by a *BusinessRule*. Likewise, the fine that must be calculated is also not described in a process diagram, but is specified in another *BusinessRule*. In contrast, the layout view does not seem to be useful to represent the existing requirements, at least in this example. However, this information might be useful when the system automates the work flow, considering the physical location of some *Items*.

A more detailed analysis of the relationships between the elements in the proposed enterprise model views and the requirements will be discussed in future works on the transformation of requirements into specifications. One step for this transformation is exactly to obtain the requirements so they can be refined.

7. Conclusion

This work proposes using an enterprise model as a requirements model, considering the scope of process automation systems. To create this meta-model, the relationship between the context, environment, and the system of the enterprise were analyzed, using the WRSPM model as a reference. Due to the fact that in the scope of process automation systems the enterprise constitutes almost all of the environment, a hypothesis was created whereby an adequate enterprise model could represent the majority of the requirements. Following a classical Administration view, from the Organization and Methods area, an enterprise meta-model was created. Finally, a small example of a library was described, where the proposed meta-model was applied and the relationship between that model and the requirements was briefly discussed.

The meta-model proposed here will be used in future works in order to transform requirements into specifications, using Model-Driven Engineering concepts. It is expected that this approach will facilitate several Requirements Engineering activities by obtaining more adequate specifications. The first step for such an

approach is to have an adequate requirements model, which is expected to be obtained with the proposed enterprise model. Nevertheless, this is an initial meta-model and so it will probably require some modification after the transformation rules have been defined. An experiment using the proposed meta-model that takes into account that approach is being executed.

References

ADDISON, M. E. **Essentials of Organization and Methods**. Heinemann, 1971.

ANDERSON, R. G. **Organisation and Methods**. 2nd edition, MacDonald And Evans, 1980.

ATKISON, C.; KÜHNE, T. **Model-Driven Development: A Metamodeling Foundation**. IEEE Software, v.20, n.5, pp.36-41, September/October 2003.

BÉZIVIN, J. **Model Driven Engineering: An Emerging Technical Space**. In: Generative and Transformational Techniques in Software Engineering, LNCS 4143, pp.36-64, 2006.

BRESCIANI, P.; PERINI, A.; GIORGINI, P.; GIUNCHIGLIA, F.; MYLOPOUYLOS, J. **Tropos: An Agent-Oriented Software Development Methodology**. In: Autonomous Agents and Multi-Agent Systems, v.8, pp.203-236, 2004.

BUBENKO JR., J.; BRASH, D.; STIRNA, J. **EKD User Guide**. Version 1.1. 1998.

CHAMPION, R. E. M.; MOORES, T. T. **Exploiting an Enterprise Model during System's Requirements Capture and Analysis**. In: International Conference on Requirements Engineering, 2., pp.208-215, 1996.

CLARK, T.; SAMMUT; P.; WILLANS, J. **Superlanguages: Developing Languages and Applications with XMF**. Ceteva, 2008.

CRUZ, T. **Sistemas, Métodos e Processos: Administrando Organizações por Meio de Processos de Negócio**. 2nd edition, Atlas, 2005 (in Portuguese).

CURY, A. **Organização e Métodos: Uma Visão Holística**. Atlas, 8th edition, 2007 (in Portuguese).

DE LA VARA, J. L.; SANCHÉZ, J.; PASTOR, O. **Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems**. In: International Conference on Advanced Information Systems Engineering, 20. LNCS 5074, pp.213–227, 2008.

DECREUS, K.; POELS, G. **Putting Business into Business Process Models**. In: International Computer Software and Applications Conference, pp.1005-1010, 2008.

DECREUS, K.; SNOECK, M.; POELS, G. **Practical Challenges for Methods Transforming i* Goal Models into Business Process Models**. In: IEEE International Requirements Engineering Conference, 17., pp.15-23, 2009.

DIETZ, J. L. G. **Deriving Use Cases from Business Process Models**. In: International Conference on Conceptual Modeling, 22., LNCS 2813, pp.131-143, 2003.

DIJKMAN, R. M.; JOOSTEN, S. M. M. **Deriving Use Case Diagrams from Business Process Models**. Technical Report TR-CTIT-02-08, Centre for Telematics and Information Technology, University of Twente, 2002.

GUNTER, C.; GUNTER, E.; JACKSON, M.; ZAVE, P. **A Reference Model for Requirements and Specifications**. IEEE Software, v.17, n.3, pp.37-43, May/June 2000.

HARRINGTON, H. J. **Business Process Improvement: the breakthrough strategy for total quality, productivity, and competitiveness**. McGraw-Hill, 1991.

JACKSON, M. **Software Requirements & Specifications: a Lexicon of Practice, Principles and Prejudices**. Addison-Wesley, 1995.

JACKSON, M. **The meaning of Requirements**. Annals of Software Engineering, v.3, pp.5-21, 1997.

KLEPPE, A.; WARMER, J.; BAST, W. **MDA Explained: The Model Driven Architecture: Practice and Promise**. Addison Wesley Professional, 2003.

LEFFINGWELL, D.; WIDRIG, D. **Managing Software Requirements: A Use Case Approach**. Addison-Wesley, 2nd edition, 2003.

LERNER, W. **Organização, Sistemas e Métodos: Solução para Renovação e Inovação Empresarial Participativa**. 5th edition, Atlas, 1992 (in Portuguese).

MAIDEN, N. **User Requirements and System Requirements**. IEEE Software, v.25, n.2, pp.90-91, March/April 2008.

MAYER, R. J.; MENZEL, C. P.; PAINTER, M. K.; DEWITTE, P. S.; BLINN, T.; PERAKATH, B. **Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report**. Interim Technical Report AL-TR-1995-XXXX, 1995.

MEYER, B. **Introduction to the Theory of Programming Languages**. Prentice Hall, 1990.

MILLER, J.; MUKERJI, J. (Eds). **MDA Guide Version 1.0.1**. omg/2003-06-01, 2003.

MOLINA, J. G.; ORTÍN, M. J.; MOROS, B.; NICOLÁS, J. TOVAL, A. **Towards Use Case and Conceptual Models through Business Modeling**. In: International Conference on Conceptual Modeling, 19., LNCS 1920, pp.281-294, 2000.

MYLOPOULOS, J.; CHUNG, L.; LIAO, S.; WANG, H.; YU, E. **Exploring Alternatives During Requirements Analysis**. IEEE Software, v.18, n.1, pp.92-96, Jan./Feb. 2001.

OLIVEIRA, D. P. R. **Sistemas, Organização & Métodos: Uma Abordagem Gerencial**. 5th edition, Atlas, 1994 (in Portuguese).

OMG – Object Management Group. **Business Process Definition MetaModel - Volume II: Process Definitions**. Version 1.0, formal/2008-11-04, 2008.

OMG – Object Management Group. **Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification**. Version 1.1, ptc/09-12-05, 2009.

OMG – Object Management Group. **Business Motivation Model**. Version 1.1, formal/2010-05-01, 2010a.

OMG – Object Management Group. **OMG Unified Modeling Language (OMG UML), Superstructure**. Version 2.3, formal/2010-05-03, 2010b.

OMG – Object Management Group. **Business Process Model and Notation (BPMN)**. Version 2.0, formal/2011-01-03, 2011.

RAMOS, I.; BERRY, D. M.; CARVALHO, J. A. **Requirements engineering for organizational transformation**. Information and Software Technology, v.47, pp.479–495, 2005.

SANTANDER, V. F. A.; CASTRO, J. F. B. **Deriving Use Cases from Organizational Modeling**. In: International Conference on Requirements Engineering, pp.32-39, 2002.

SEIDEWITZ, E. **What Models Mean**. IEEE Software, v.20, n.5, pp.26-31, 2003.

SIKDAR, S.; DAS, O. **Stakeholder Appropriate Requirements Development Approach**. In: International Advance Computing Conference, p.1670-1674, 2009.

THE ECLIPSE FOUNDATION. **Eclipse IDE**. Version 3.6.1, 2010.

THE OPEN GROUP. **TOGAF Version 9**. 2009. Available at: <http://www.opengroup.org/togaf/>.

THEVENET, L.; SALINESI, C. **Aligning IS to Organization's Strategy: The INSTAL Method**. In: Conference on Advanced Information Systems, LNCS 4495, pp.203–217, 2007.

VAN LAMSWEERDE, **A. Requirements Engineering: From System Goals to UML Models to Software Specifications**. John Wiley & Sons, 2009.

VERNADAT, F. B. **Enterprise Modeling and Integration: Principles and Applications**. Chapamn & Hall, 1996

YU, E. S. **Modelling Strategic Relationships for Process Reengineering**. 166p. Doctoral Thesis – Department of Computer Science, University of Toronto, 1995.

YU, E.; MYLOPOULOS, J.; LESPERANCE, Y. **AI Models for Business Process Reengineering**. IEEE Expert, pp.16-23, 1996.

ZAVE, P.; JACKSON, M. **Four Dark Corners of Requirements Engineering**. ACM Transactions on Software Engineering and Methodology, v.6, n.1, pp.1-30, jan. 1997.