Bridging the Gap between a Set of Interrelated Business Process Models and Software Models

Estrela F. Cruz[1,2,*], Ricardo J. Machado[2] and Maribel Y. Santos[2]
[1]*Instituto Politécnico de Viana do Castelo, Viana do Castelo, Portugal*
[2]*Centro ALGORITMI, Escola de Engenharia, Universidade do Minho, Guimarães, Portugal*

Keywords: Business Process Modeling, BPMN, Use Case Model, UML, Requirements Elicitation.

Abstract: A business process model identifies the activities, resources and data involved in the creation of a product or service, having lots of useful information for starting to develop a supporting software system. With regard to software development, one of the most difficult and crucial activities is the identification of system functional requirements. A popular way to capture and describe those requirements is through UML use case models. Usually an organization deals with several business processes. As a consequence, a software product does not usually support only one business process, but rather a set of business processes. This paper presents an approach that allows aggregating in one use case model all the information that can be extracted from the set of business process models that will be supported by the software under development. The generated use case model serves as a basis for the software development process, helping reducing time and effort spent in requirements elicitation. The approach also helps to ensure the alignment between business and software, and enables traceability between business processes and the corresponding elements in software models.

1 INTRODUCTION

Business Process Management (BPM) is being increasingly used by organizations as a means to improve the quality of their products or services, the efficiency of their processes and productivity and increasing the benefits for their stakeholders. By this way, BPM is becoming increasingly important to organizations. BPM includes methods, techniques, and tools to support the design, enactment, management, and analysis of operational business processes (van der Aalst, 2004). A business process is a set of interrelated activities that are executed by one, or several, organizations working together to achieve a common business purpose (Ko, 2009). Among the various existing modeling languages, we opted for the Business Process Model and Notation (BPMN), currently in version 2.0 (OMG, 2011), because it is a widespread OMG standard that is very well accepted and actually used in companies and organizations (Recker, 2008). Moreover, it is a complete language that allows creating a detailed business process model (OMG, 2011).

Information systems researchers and professionals have recognized that understanding a business process is the key to identify the user needs of the software that supports it (Shishkov et al., 2002; Ko, 2009). In fact, one of the main software quality objectives that needs to be addressed is to ensure that a software product meets the business needs (Jalote, 2008). For that, the software product requirements need to be aligned with the business needs, both in terms of business processes and in terms of the informational entities that those processes deal with. This drives us to the question: "Is it possible to systematically derive a use case model from a set of interrelated business process models?"

Requirements elicitation is a key step in the software development process. Nevertheless it is a time-consuming step that can take several months (or even years) to complete and usually involves many resources (Cockburn, 2001). Use case models aim to capture and describe the functional requirements of a system (Hull et al., 2011). A use case model is a set of use case diagrams and the corresponding use case descriptions (Bittner and Spence, 2003). The use case diagrams enable to perceive the need of describing the system behavior in response to messages received from outside the system (i.e., from its actors and external systems) (Hull et al., 2011).

*This work has been supported by FCT - *Fundação para a Ciência e Tecnologia* in the scope of the project: PEst-UID/CEC/00319/2013.

Ferreira Cruz E., Machado R. and Santos M..
Bridging the Gap between a Set of Interrelated Business Process Models and Software Models.
DOI: 10.5220/0005378103380345
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 338-345
ISBN: 978-989-758-097-0

Approaches such as BPEL (Business Process Execution Language) allow the execution of business processes in a service-oriented perspective by integrating enterprise applications (Liang et al., 2008). Such approaches, however, require the existence of services, which could be internal or external to the organization, whose orchestration can be done using BPEL. The aim of this approach is to generate models that can be used as basis to the development of software that supports the business processes. To plan and design a suitable software supporting system, first it is necessary to know the main requirements that must be supported by the software under development. The approach presented in this paper intends to identify the functional requirements (representing them as a use case model) based on the set of interrelated business processes. Shishkov *et al.* state that deriving use case models from business analysis models would be useful, since both reflect behavior within business/software systems (Shishkov et al., 2002).

This paper further complements and improves previous work, presented by the same authors, on bridging the gap between business process models and software models. The approach presented in (Cruz et al., 2014a) derives a use case model from one business process model. However, in a real situation a software product does not usually support only one business process, but rather a set of business processes. In the approach presented herein we intend to improve the work presented in (Cruz et al., 2014a) extending it in order to treat sub-processes without losing information and to aggregate and merge the information we have in a set of interrelated business processes into one integrated use case model.

The remainder of this paper is structured as follows. Next section summarizes two fundamental approaches, which are used in the approach described herein, and presents some related work. The proposed approach is presented in section 3. The application of the proposed approach is illustrated through a demonstration case in section 4. Finally, conclusions and some remarks to future work are presented in section 5.

2 PREVIOUS AND RELATED WORK

Previous work by the same authors (Cruz et al., 2014a), summarized next, allows generating a use case model, including descriptions, from one private BPMN business process model. Nevertheless, when sub-processes are involved, the approach may lose some structuring information (Cruz et al., 2014a) and most of all, the approach deals with a single process. Usually, a software product does not support only one business process, but rather a large number of business processes. In this case, we have to group in one use case model all the information we have in the set of business process models that will be supported by the software under development.

The approach described in this paper, and presented in section 3, starts with high abstraction level use cases, each corresponding to a business process, and refines them (by decomposition) in order to obtain detailed use cases, useful to software development. To do that we are using the decomposition triangle approach (Cruz et al., 2014b) (summarized next) that refines a use case into a more detailed use case model.

From one Business Process Model to a Use Case Model

An approach to generate a use case model, including descriptions, from a single private business process diagram (modeled in BPMN) has been presented in (Cruz et al., 2014a). The approach comprises the definition of a set of rules to generate a use case diagram in which each activity in the BPMN model gives origin to a use case, and a participant gives origin to an actor in use case model. The relationship between actors and use cases is derived from the existing relations between the corresponding participant and activities. The actor that is derived from a pool (or lane) is related with the use cases that are derived from the activities belonging to that pool (or lane). The actor that is derived from a participant that exchanges messages with an activity is related with the use case derived from the activity.

All existing information (data involved in the process, decisions that have to be made, exchanged messages and so on) that cannot be represented as an actor or as a use case is depicted in the use cases descriptions. To describe a use case, a template that simplifies the Cockburn's template for use cases is used. Based on a private business process model the approach is able to identify the use case names and the related actors. Depending on the incoming and outgoing connections from the corresponding activity, it is also able to identify each use case's pre-conditions, post-conditions, trigger and the main scenario.

The Decomposition Triangle Approach

An approach, named as Decomposition Triangle, to decompose and refine use cases has been presented in (Cruz et al., 2014b). The approach starts with high

abstraction level use cases and ends with very detailed level use cases. An extension to the UML2.5 is also proposed for accommodating a new refinement relation from a use case to a use case (sub-)model (Cruz et al., 2014b).

The approach is iterative and incremental and starts by identifying the system scope and the actors involved. Then, the system's main functionalities are identified and represented as use cases in the use case model *level 1*. Based on the use case descriptions and stakeholder's feedback, each use case can be decomposed in a use case model represented on the next decomposition level, decreasing this way the abstraction level and increasing the detail level. The process ends when the CRUD (Create, Retrieve, Update and Delete) operations are identified. The use cases identified are numbered using a leveled numbering.

Related Work

Lubke and Schneider (Lubke and Schneider, 2008) propose an approach to generate a business process model (modeled in BPMN) from a UML use case diagram. The authors justify the need for this approach with the increasing number of use cases and with the possibility of losing the execution order of the various use cases (Lubke and Schneider, 2008).

Other authors propose approaches to derive use cases from business process models. Some of the existing approaches are presented next.

Dijkman and Joosten propose an approach that maps a business process model (modeled using the UML Activity Diagram) into use case diagrams (Dijkman and Joosten, 2002b). They also proposed an algorithm to derive a use case diagram from a business process modeled as activity diagrams (Dijkman and Joosten, 2002a). To do so, Dijkman and Joosten start by defining the activity diagram and the use case diagram meta-models. Then, the authors establish a relation between the "role" from the activity diagram and the "actor" in a use case diagram and a "step" (a sequence of tasks) from the activity diagram originates a "use case" in a use case diagram (Dijkman and Joosten, 2002a).

Rodríguez *et al.* propose a systematic approach to derive a use case diagram from a UML activity diagram (Rodríguez et al., 2008) and another to derive a use case diagram from a BPMN model (Rodríguez et al., 2007). In the latter approach, the transformation is guided by a set of QVT (Query View Transform) rules and checklists. In a summarized way, in the Rodriguez *et al.* approach, a participant is mapped to an actor in the use case diagram and an activity gives origin to a use case.

Cockburn (Cockburn, 2001) distinguishes different use case abstraction levels. The low detail level use cases, and the very detailed use cases with a clear intention. Cockburn states that low detail level use cases are not trustable as functional requirements for the system being built (Cockburn, 2001). From another point of view, Cockburn categorizes use cases as business use cases and system use cases (Cockburn, 2001). Cockburn sees business use cases as low detail level (high abstraction) use cases and system uses cases as high detail level (low abstraction) use cases and advises the use case writers to start with the business use cases and "unfold" them continuously until they become system use cases (Cockburn, 2001).

The approach we are presenting herein is aligned with Cockburn's points of view, helping in transforming business process models (low detail level use cases) into system level functional requirements (high detail level use cases), and helping in keeping track of the transformations of the use cases.

All surveyed existing approaches obtain a use case diagram based on a single business process model. None of the surveyed approaches aggregates a set of business processes models in one use case model including the use case descriptions. But, typically, in a real situation, a software product does not support only one process, but a reasonable set of processes. In order to generate useful use case model it will be necessary to consider the set of business process models that will be supported by the software under development.

3 DERIVING A USE CASE MODEL FROM A SET OF INTERRELATED BUSINESS PROCESS MODELS

In the approach presented herein we intend to improve the work presented in (Cruz et al., 2014a) extending it in order to treat sub-processes without losing information and to aggregate and merge the information we have in a set of interrelated business processes into one integrated use case model.

The set of business processes, belonging to an organization, being supported by the software under development must be grouped in a single use case model because a software development team needs to understand the system context and scope before starting to plan and design a solution. Following this reason, first it is needed to identify and specify which business processes are to be supported by the software under development.

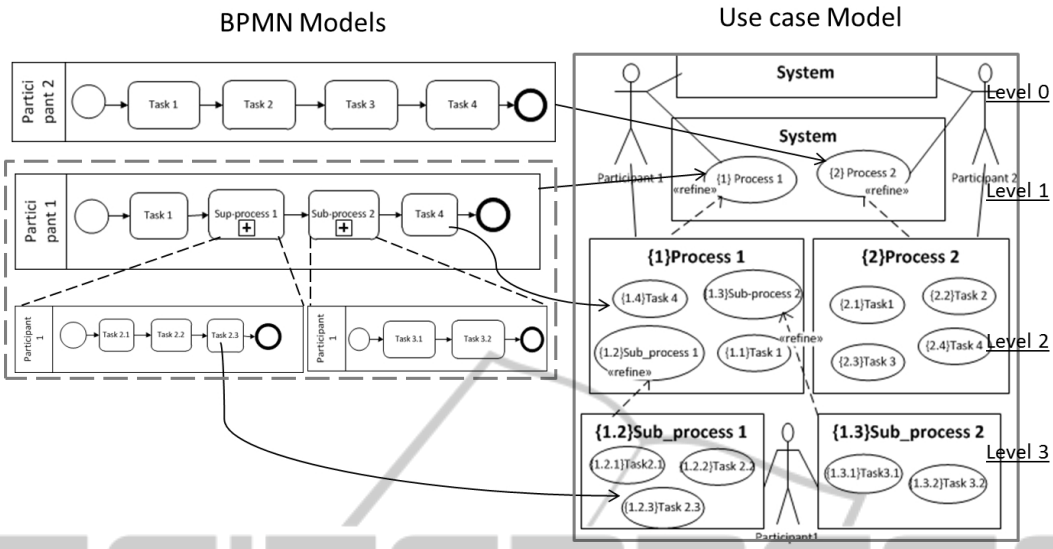

Figure 1: A generic decomposition scheme.

A use case model can be created with a high abstraction level or low abstraction level (Cockburn, 2001; Cruz et al., 2014b). The approach we are presenting here starts with high abstraction level use cases and ends with lower abstraction level use cases. The approach starts by grouping all processes that will be supported by the software under development in one use case diagram where each process is represented as a use case. Each use case is then refined and decomposed in a use case model as described in (Cruz et al., 2014b). All identified use cases are numbered using the *tag=value* UML mechanism. A generic scheme of applying the decomposition is shown in Figure 1.

Similarly to what happens in (Cruz et al., 2014b), the approach represents the use case models in different abstraction levels:

- ***Level 0:*** At this level, the system scope and frontier must be identified as well as all the actors involved, so the set of business processes that will be supported by the software under development must be identified. Each participant in a business process model (represented as a pool or a lane) is transformed in an actor (with the same name) in the use case diagram (see Figure 1, level 0). The subdivision of a pool in several lanes, or a lane in other lanes originates an actor's hierarchy (Cruz et al., 2014a). Thus, at this level all participants involved in the set of business processes are represented, as actors, in the actor's diagram.

  Usually a participant is involved in several business processes belonging to an organization. When participants with the same name are involved in more than one business process we assume that they represent the same participant, and, as consequence, they will be represented by the same actor in the use case model.

- ***Level 1:*** At this level, the first use case model is created with the highest abstraction level where each top level business process is transformed into a use case, in the use case model (a business use case) (see Figure 1, level 1). Each use case, representing a business process, is related with the corresponding actors representing the participants involved in the process. The use cases are numbered sequentially. The use case description is a general overview of the process it addresses.

  Monsalve *et al* state that a business process model does not allow identifying business process goals or main objectives, but that is important to software requirements elicitation (Monsalve et al., 2012). At this level, the use case description can be used to represent the business process goal and objective helping to understand the business process purpose. That can be accomplished by talking/discussing with stakeholders since at this software development process' stage the stakeholders are still actively involved in the process.

- ***Level 2:*** At this level, each process (represented as a use case in level 1) is mapped to a use case model following the rules described in (Cruz et al., 2014a). Basically one activity from a business process is transformed into a use case and each participant is transformed into an actor. All incoming and outgoing connection flows from the activity originate a NL sentence in the description of the use case that represents the activity. Each generated use case model in *level 2* refines and

decomposes a corresponding use case in *level 1* (see Figure 1, level 2). The number of use case models, at this level, is the same as the number of use cases existing in the diagram at *level 1*. The use cases identified are numbered using a leveled numbering.

- ***Level (i+1)***, *(i≥2)* **:** At this level, each use case that represents a sub-process in *level i* is decomposed and refined in a use case model in *level (i+1)* (see Figure 1, level 3). BPMN has two types of activities: a task (atomic activity) and a sub-process (OMG, 2011). A sub-process is a process, so in this case, the use case that represents the activity can be decomposed and mapped to a use case model. The decomposition ends when all use cases representing processes or sub-processes are refined.

In the presented approach, refining a use case means detailing all activities involved in the corresponding process, including all resources and/or data that are consumed and produced, messages exchanged, decisions that have to be taken, events that can occur, etc.

The decomposition results in a tree structure where the leaf nodes represent the tasks and the non-leaf nodes represent the processes and sub-processes. The decomposition tree has high abstraction level use cases at *level 1*. The abstraction level decreases in every use case decomposition. The approach allows to relate use cases belonging to different abstraction levels allowing to drill down and up between different abstraction levels. This way, the approach allows tracing back from requirements to the business processes and from the business processes to the corresponding requirements.

4 DEMONSTRATION CASE

In this section we use, as a demonstration case, a very well-known example of a School Library System where some of the business process more commonly used have been selected to present here: *Register User*, *Lend a Book*, *Purchase a Book*, *Return a Book* and *Renew Loan*. The *Return a Book* business process model includes the sub-process, *Penalty treatment*.

Figure 2 shows the use case diagram level 0 (actors diagram) with four actors, which were derived from the business process participants. The actors are also depicted in diagram level 1 as seen in Figure 3.

In the *Purchase a Book* business process model (represented in Figure 4) the participants involved are

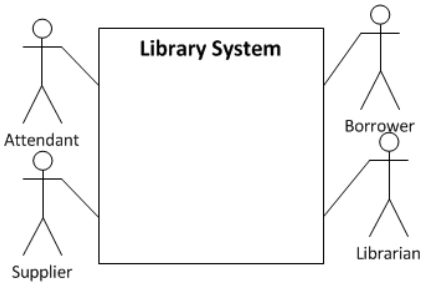

Figure 2: Actors diagram (level 0).

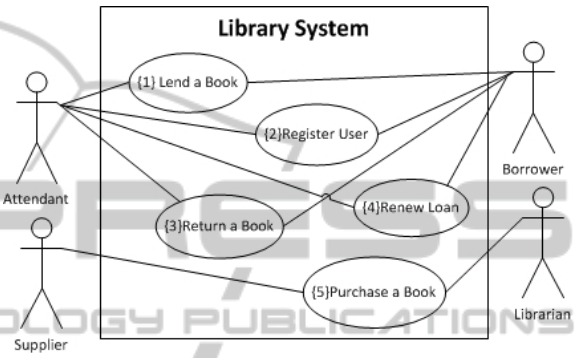

Figure 3: Use Case diagram (level 1).

Librarian and *Supplier*. Following the rules presented previously, *Librarian* and *Supplier* must be represented as actors in the actors diagram *level 0* and in use case diagram *level 1* (see Figure 3).

In *Lend a Book* business process model (Figure 5) two participants are identified, *Borrower* and *Attendant*. The two corresponding actors (with the same name) will be also represented in the actors diagram and in the use case diagram *level 1* (see Figure 3).

In *Return a Book* business process (Figure 6), the participants involved are *Borrower* and *Attendant* already identified in *Lend a Book* business process model. So, the actors belonging to the actors diagram and use case diagram *level 1* remain the same.

For limitations of space, the *Register User* and *Renew Loan* business process models are not presented here, but in both processes the participants involved are the same *Borrower* and *Attendant*. The corresponding actors are already represented in the actors diagram and in the use case diagram level 1.

Summarizing, and looking at the business processes we have selected for our demonstration case, we can see that *Librarian* and *Supplier* are the participants involved in *Purchase a Book* business process, *Borrower* and *Attendant* are the participants involved in all the other processes. Thus, these four participants are represented as actors in the actors diagram (level 0), as seen in Figure 2.

In the use case diagram level 1 (refer to Figure 3),

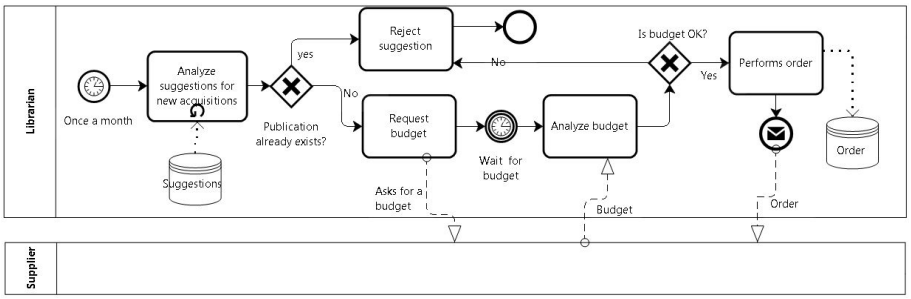

Figure 4: Purchase a Book business process model.

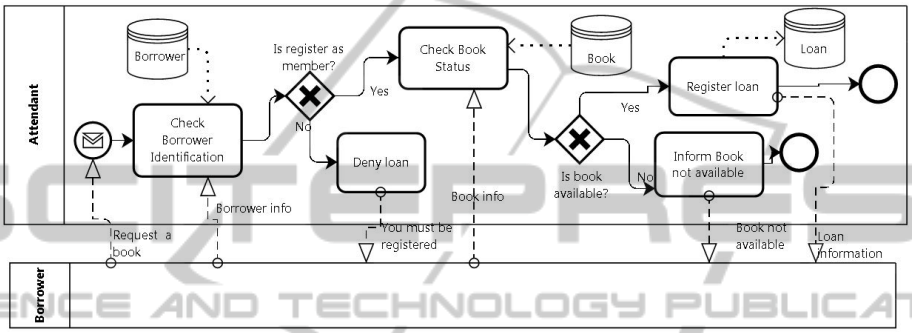

Figure 5: Lend a Book business process model.

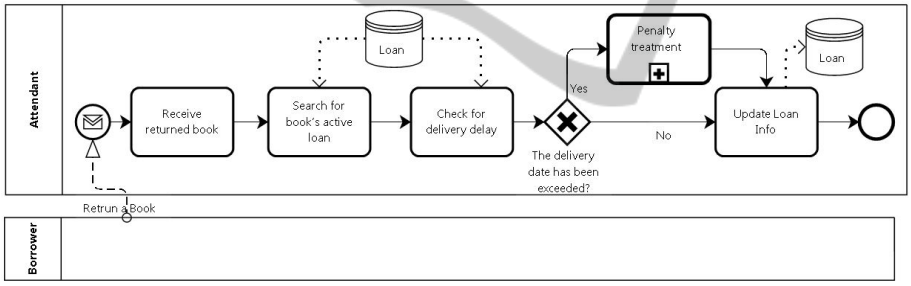

Figure 6: Return a Book business process model.

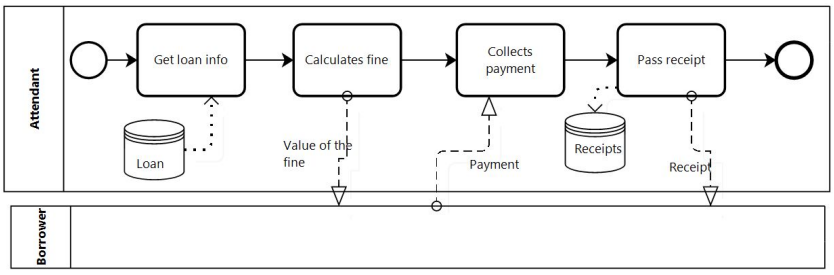

Figure 7: Penalty Treatment business process model.

each selected business process is represented as a use case connected with the actors that represent the corresponding participants. So, *Purchase a Book* business process is related to actors *Librarian* and *Supplier*. *Lend a Book*, *Return a Book*, *Renew Loan* and *Register User* use cases are related with the actors *Borrower* and *Attendant*. The use cases are numbered sequentially. Each use case, which represents a process, is then detailed in a use case model at level 2, as we can see in Figure 8.

When an actor is related with all use cases in a use case diagram, the established association between the actor and the use cases is represented by a single link to the diagram boundary, to simplify the repre-

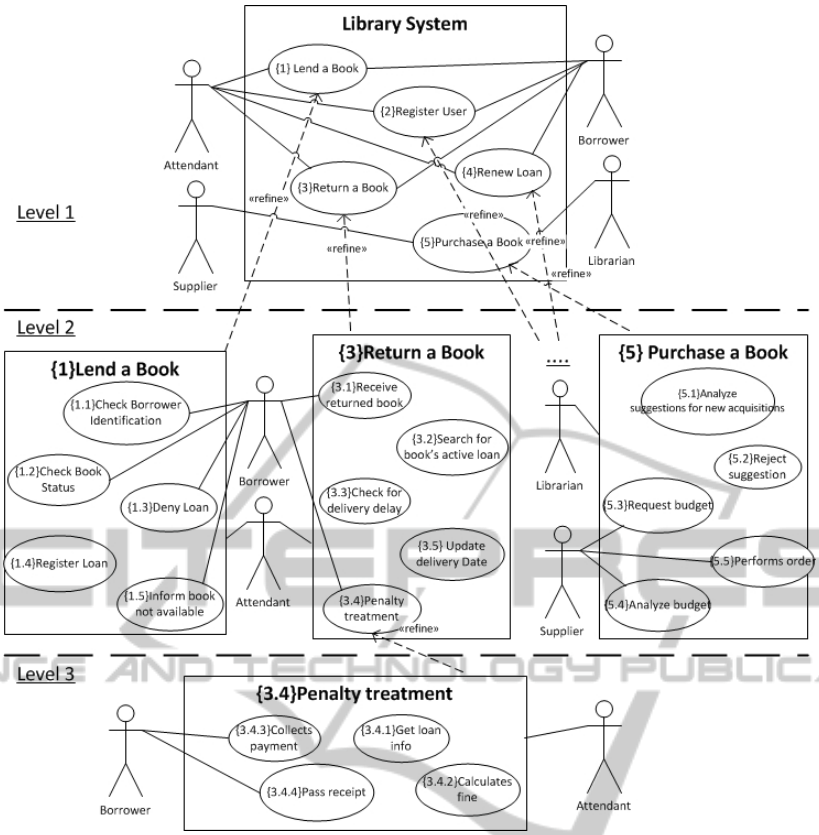

Figure 8: Complete Use Case model (level 3).

sentation. If an actor is associated to only some of the use cases that belong to the use case diagram, the link is established between the actor and the related use cases.

By analyzing the *Return a Book* business process model (refer to Figure 6), we can see that the process comprises five activities: *Receive returned book*, *Search for book's active loan*, *Check for delivery delay*, *Update Loan Info* and *Penalty treatment*, each one giving origin to a use case in the use case model that refines the *Return a Book* use case (level 2). All activities belong to the *Attendant* pool, so all use cases representing the activities are related with the *Attendant* actor. *Receive returned book* activity receives a message from the *Borrower* participant and *Penalty treatment* activity also exchange messages with the *Borrower* participant, so the *Borrower* actor is related with both corresponding use cases. Similar line of reasoning can be applied to explain the *Lend a Book* (refer to Figure 5) and *Purchase a Book* (Figure 4) business process models transformation.

The *Penalty Treatment* use case represents a subprocess (represented in Figure 7), so it can be represented as a use case model in the next level (level 3), detailing and refining the use case (level 2) as we can see in Figure 8.

5 CONCLUSIONS AND FUTURE WORK

The approach presented in this paper allows concluding that is it is possible to systematically derive a complete data model from a set of interrelated business process models, helping to ensure that the software requirements really meet business needs.

The approach starts by identifying the set of business processes that will be supported by the software under development, identifying the system scope. Then a use case model, divided in several abstraction levels, is created based on a set of identified business process models. In level 0, all actors and all actors' hierarchies are identified, representing the involved participants. In level 1, the highest abstraction level use cases are represented in the use case model, each one representing a business process being supported by the software system under development. Each use case is then decomposed and refined in a use case model (level 2). All existing information in a private

business process model is transformed to information in the use case model. Most of the information is transcribed to use case descriptions. The approach ends when all use cases representing processes and subprocesses are decomposed into atomic activities, each one being represented as a use case.

The decomposition process generates a use case tree structure, which enables one to drill down and easily trace any use case back to its more abstract base use case and corresponding business process, allowing traceability between business processes and software representations. The created use case model will serve as a basis to the development of the software that will support the business. At the same time, business and software modeling efforts can be joined together, reducing the analysis time and avoiding forgetting functional requirements.

Generating a use case model joining the information from a set of business process models allows us to use existing methods, techniques and tools to generate other software models from use case models. One of those methods is the 4SRS (4-Step Rule Set), which generates a logical architecture and corresponding class diagrams from user requirements, represented as use cases (Santos and Machado, 2010).

As future work, we intend to apply this approach, integrated with the 4SRS, in a real industrial scenario which complexity and dimension will benefit from a systematic approach to the generation of the use case model and other software models.

REFERENCES

Bittner, K. and Spence, I. (2003). *Applying use cases: a practical guide*. Pearson Education inc.

Cockburn, A. (2001). *Writing Effective Use Cases*. Addison Wesley.

Cruz, E. F., Machado, R. J., and Santos, M. Y. (2014a). From business process models to use case models: A systematic approach. In Aveiro, D., Tribolet, J., and Gouveia, D., editors, *Advances in Enterprise Engineering VIII*, volume 174 of *LNBIP*, pages 167–181. Springer International Publishing.

Cruz, E. F., Machado, R. J., and Santos, M. Y. (2014b). On the decomposition of use cases for the refinement of software requirements. In *14th ICCSA, IEEE Computer Society Press*, pages 237–240.

Dijkman, R. M. and Joosten, S. M. (2002a). An algorithm to derive use cases from business processes. In *6th Int.Conf. on SEA*, pages 679–684.

Dijkman, R. M. and Joosten, S. M. (2002b). Deriving use case diagrams from business process models. Technical report, CTIT Technical Report, The Netherlands.

Hull, E., Jackson, K., and Dick, J. (2011). *Requirements Engineering*. Springer.

Jalote, P. (2008). *A concise Introduction to Software Engineering*. Springer.

Ko, R. K. L. (2009). A computer scientist's introductory guide to business process management (bpm). *Crossroads*, 15:4:11–4:18.

Liang, Y., Tian, J., Hu, S., Song, Y., and Zhang, Y. (2008). A template-based approach for automatic mapping between business process and BPEL process. *Wuhan University Journal of Natural Sciences*, 13:445–449.

Lubke, D. and Schneider, K. (2008). Visualizing use case sets as BPMN processes. In *Requirements Engineering Visualization*.

Monsalve, C., April, A., and Abran, A. (2012). On the expressiveness of business process modeling notations for software requirements elicitation. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pages 3132–3137.

OMG (2011). Business process model and notation (BPMN), version 2.0. Technical report, OMG.

Recker, J. C. (2008). BPMN Modeling – Who, Where, How and Why. *BPTrends*, 5(3):1–8.

Rodríguez, A., Fernández-Medina, E., and Piattini, M. (2007). Towards CIM to PIM transformation: From secure business processes defined in BPMN to use-cases. In *B. Process Management*, pages 408–415.

Rodríguez, A., Fernández-Medina, E., and Piattini, M. (2008). Towards obtaining analysis-level class and use case diagrams from business process models. In *Advances in Conceptual Modeling Challenges and Opportunities*, volume 5232 of *LNCS*, pages 103–112. Springer Berlin / Heidelberg.

Santos, M. Y. and Machado, R. J. (2010). On the derivation of class diagrams from use cases and logical software architectures. In *2010 Fifth ICSEA*.

Shishkov, B., Xie, Z., Liu, K., and Dietz, J. L. (2002). Using norm analysis to derive use cases from business processes. In *Proceedings of the 5th Workshop On Organizational Semiotics*.

van der Aalst, W. (2004). Business process management demystified: A tutorial on models, systems and standards for workflow management. In Desel, J., Reisig, W., and Rozenberg, G., editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 1–65. Springer Berlin / Heidelberg.