

From Business Process Models to Use Case Models: A Systematic Approach

Estrela Ferreira Cruz^{1,2}, Ricardo J. Machado², and Maribel Yasmina Santos^{2,*}

¹ Instituto Politécnico de Viana do Castelo, Portugal
estrela.cruz@estg.ipv.pt

² Centro ALGORITMI, Escola de Engenharia,
Universidade do Minho, Guimarães, Portugal
{rmac,maribel}@dsi.uminho.pt

Abstract. One of the most difficult, and crucial, activities in software development is the identification of system functional requirements. A popular way to capture and describe those requirements is through UML use case models. 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 developing a supporting software system. During system analysis, most of this information must be incorporated into use case descriptions. This paper proposes an approach to support the construction of use case models based on business process models. The proposed approach obtains a complete use case model, including the identification of actors, use cases and the corresponding descriptions, which are created from a set of predefined natural language sentences mapped from BPMN model elements.

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

1 Introduction

Markets' globalization and the constant increase of competition between companies demand constant changes in organizations in order to adapt themselves to new circumstances and to implement new strategies. Organizations need to have a clear notion of their internal processes in order to increase their efficiency and the quality of their products or services, increasing the benefits for their stakeholders. For this reason, many organizations adopt a business process management (BPM) approach. BPM includes methods, techniques, and tools to support the design, enactment, management, and analysis of operational business processes [1]. 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 [2]. Among the various existing modeling languages, we opted for the Business Process Model and Notation (BPMN), currently in version 2.0 [3], because it is a widespread OMG standard that is actually used both in academia and in organizations.

* This work has been supported by FCT - Fundação para a Ciência e Tecnologia within the Project Scope: PEst-OE/EEI/UI0319/2014.

If on one hand the business process management and modeling are increasing their relevance, on the other hand the software development teams still have serious difficulties in performing elicitation and defining the applications requirements [4]. In fact, one of the main software quality objectives is to assure that a software product meets the business needs [4]. 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: “Can the existing model information about business processes be used as a basis for modeling the software applications that support that business?”

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 [5,6]. However, the tasks of business process analysis and software development are managed by different groups of people and commonly use different languages.

Requirement elicitation is, indeed, a key step in the software development process. Use case models aim to capture and describe the functional requirements of a system [7]. Dietz says that the use cases strong point is that once they are identified, the development of the software application goes well [8]. The weak point is the identification of use cases themselves. Shishkov *et al.* states that deriving use case models from business analysis models would be useful, since both reflect behavior within business/software systems [6].

A use case model is a set of use case diagrams and the corresponding use case descriptions [9]. 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) [10].

In this paper, we present an approach to obtain a complete use case model based on a business process model. All information existing in a BPMN model that cannot be represented as an actor or as a use case will be depicted as textual use case description. Use case descriptions are, commonly, specified in Natural Language (NL) [11,12]. As Fantechi *et al.* say NL is easy to understand but, at the same time, could be ambiguous, redundant and with omissions [11]. However, the generated descriptions are a set of controlled sentences previously defined in NL.

The remainder of this paper is structured as follows. In the next section, BPMN and basic concepts of use case models are introduced and some related work is presented. Section 3 describes our approach for use case model creation and presents its application to an example. Finally, conclusions and some remarks to future work are presented.

2 Background

2.1 The BPMN Language

Business process management focus its attention on designing and documenting business processes, in order to describe which activities are performed and

the dependencies between them [13]. The BPMN basic process models can be grouped into two types of processes [3]:

- **Private Business Processes** - A private process is a process internal to a specific organization. Each private process is represented within a Pool. The process flow must be in one pool and should never cross the boundaries of that Pool. The interaction between distinct private Business Processes can be represented by incoming and outgoing messages.
- **Public Processes** - A public process represents the interactions between a private Business Process and other Processes or Participants. Only activities that are used to communicate with the other participants must be included in the public process.

The BPMN's diagrams use a set of graphical objects that can be grouped into five basic categories [3]:

- **Flow Objects** - are the main graphical elements to define the behavior of a Business Process. There are three kinds of Flow Objects: Events, Activities and Gateways.
- **Data** - represent the data involved in the process. Data that flows through a process is represented by *data objects*. Persistent data can be represented by *data stores*. Data objects and data stores are exclusively used in private process diagrams [3].
- **Connecting Objects** - model the connection between the several process elements. There are four types of connecting objects: Sequence Flows, Message Flows, Associations and Data Associations.
- **Swimlanes** - represent the participants in the process. A participant is a person, or something, involved in the process. Participants in the process can be grouped into pools or, more particularly, in Lanes. A pool can be divided into several Lanes, for example, to represent the different departments of an organization involved in the process.
- **Artifacts** - are used to provide additional information to the process, such as a note (“Text Annotation”).

During a process execution, resources and/or data are consumed and produced. The transmission of the data created or used during a process execution can be represented by *Messages* or *Data Associations*.

The following subsection addresses use case models.

2.2 Use Case Model

Booch *et al.* say that use case models, when defined by Ivar Jacobson, aimed to describe the behavior of the system from the users point of view [14]. So, it is expected that a use case model specifies what a system is supposed to do [15]. In [15] a use case is defined as a *behavioral classifier* that represents a declaration of a set of offered behaviors. Each use case specifies some behavior, possibly including variants, which the subject can perform in collaboration with one or more actors.

A use case model should identify the system boundaries (depicted as a rectangle) and the actors, which are represented by a “stickman” icon outside the system boundaries [7,15]. An actor is someone or something that interacts with the system [15]. So, an actor is always related to one or more use cases. A use case is graphically represented by an ellipse and contains a brief description of the action [9]. A use case diagram is composed by actors and use cases. Each use case shall have an associated description. There are some alternatives that can be used to describe a use case, like informal text, numbered steps, pseudo-code, among others [12]. Cockburn proposes a basic use case descriptions template that includes the use case name, actors, scope, context, pre-conditions, primary success scenario, alternate scenarios, amongst others [12].

2.3 Existing Approaches

It is recognized that the software that supports the business must be aligned with the business processes [16]. Therefore, it is natural to try an approximation between business process modeling and software modeling. Requirements elicitation is usually the first phase on a software development process. Several authors already 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 [17]. They also proposed an algorithm to derive a use case diagram from a business process modeled as activity diagrams [18]. 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 [18].

Rodriguez *et al.* propose a systematic approach to derive a use case diagram from a UML activity diagram [19] and another to derive a use case diagram from a BPMN model [20]. In the latter approach, the transformation is guided by a set of QVT (Query View Transform) rules and checklists. In a summarized way, in Rodriguez *et al.* approach, a participant is mapped to an actor in the use case diagram; an activity in the BPMN model gives origin to a use case.

All surveyed existing approaches obtain a use case diagram based on a business process model, but no one presents a proposal for obtaining the use cases description. Nevertheless, the use cases descriptions are one of the most important components of the use case model [12,21]. Moreover, without descriptions most information presented in a business process model will be lost when generating the use case diagram from a business process model.

Cockburn emphasizes the use case descriptions. In Cockburn’s opinion the use case writers should spend their time and effort on use case descriptions [12]. The use case descriptions can specify all information needed. But, how should the use cases be written? Cockburn advises the use case writers to use sentences with a simple structure, which should be “easy to read and follow” [12] and describes a semi-formal structure to use cases description.

The CREWS (Co-operative Requirements Engineering With Scenarios) team proposes two sets of guidelines to be used on use case descriptions: six guidelines related to style and eight related to content [22]. Karl Cox also presents a set of structure guidelines for use case descriptions [23]. More exactly he proposes the *CP Use Case Writing Rules*, a small set of guidelines derived from the 7 C's (Coverage, Cogent, Coherence of logic, Consistent abstraction, Consistent Structure, Consistent Grammar, Consideration of alternatives) [23].

Comparing CREWS and CP guidelines, the CP guidelines number is smaller and intends to be easier to apply than CREW guidelines [24]. Both provide improvements on use case descriptions quality [24] and subsequently improve the understanding between stakeholders.

The next section describes our approach to obtaining the use case model from a business process model.

3 The Proposed Approach

Graphically a use case diagram is very simple because it only involves actors and use cases (stickman's and ellipses with a brief description). A BPMN process diagram is graphically more complex because it involves lots of graphical elements (activities, events, gateways, data objects, pools, etc.). However a use case model can represent as much information as a BPMN model, but most of the information must be embodied in use case descriptions. So, the approach presented here is specially focused on use case descriptions for which we present a template.

The approach is divided in two main parts. First we present a set of rules to obtain a use case diagram from a BPMN model. Then we address the rules to derive the description of the uses cases previously identified.

3.1 Use Case Diagram Generation

The presented approach is based on the private business process, where messages exchanged with other participants, or business partners, shall be represented. The proposed approach is based on the following considerations:

- The information about the participants in the process is relevant to the process, so all participants involved in messages exchange must be represented.
- An activity represents some work performed within a business process. An activity may be atomic, usually represented as a task, or non-atomic, represented as a sub-process. To avoid information loss during the application of the proposed approach, the sub-processes must be expanded.
- A manual task is a task performed without any information technology involvement [25]. Nevertheless, the information about the task execution, like start and ending time or amount of resources produced and consumed, can be useful to the process monitoring to support and evaluate future decisions or improvements.

We agree with Rodriguez *et al.* on mapping a participant to an actor and one activity to a use case [20]. Accordingly, the rules to generate the use case diagram are explained below:

- R1: A role played by a participant (represented by a lane or a pool) must be represented by an actor in the use case diagram. The actor name is the participant name.
- R2: A lane can be the sub-division of a pool or a sub-division of another lane. These subdivisions form the actors' hierarchy:
 - If the lane is a sub-division of a pool then the actor that represents the lane is a specialization of the actor that represents the pool;
 - If the lane is a sub-division of another lane then the actor that represents the internal lane is a specialization of the actor that represents that lane.
- R4: Each activity will be represented as a use case in the use case diagram. The use case name (brief description of the action) is the activity name.
- R5: An actor that represents a pool (or a lane) is related with all use cases representing the activities that belong to the pool (or lane).
- R6: The actor that represents the participant that sends (or receives) a message to an activity is related to the use case that represents that activity.

Next subsection applies the described rules to the Nobel Prize example.

3.2 Nobel Prize Example

The diagram shown in Figure 1, represents the Nobel Prize BPMN Process Diagram. The presented BPMN model comprises ten activities, consequently (by rule R4 above) there will be ten use cases on the generated use case diagram. Four pools are involved in the process: *Nobel Committee*, *Nominators*, *Expert* and *Nobel Assembly*. By R1 the obtained use case diagram will have four actors with the corresponding names. The obtained Nobel Prize use case diagram is shown in Figure 2.

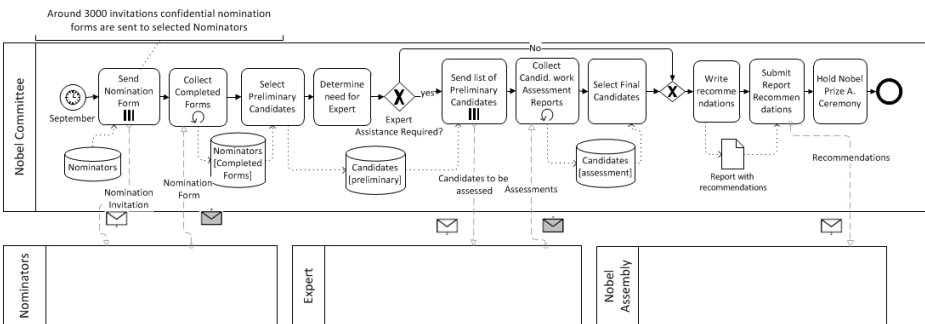


Fig. 1. The Nobel Prize Process Diagram (adapted from [26])

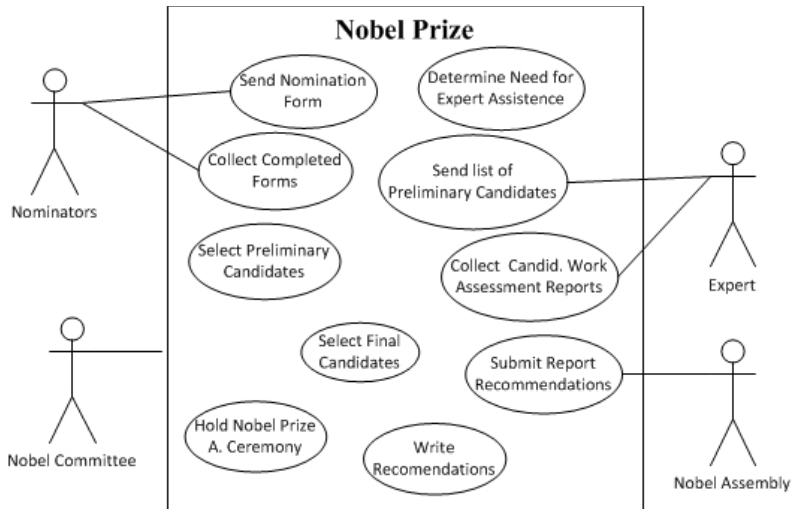


Fig. 2. The Nobel Prize Use Case Diagram

As can be seen in Figure 1, all activities are performed by *Nobel Committee* participant, so, by R5, all use cases are related with *Nobel Committee* actor. The *Nominators* participant sends a message to *Send Nomination Form* activity, so, by R6, the *Nominators* actor is related with the *Send Nomination Form* use case. The *Collect Completed Forms* activity receives a message from the *Nominators* pool, so, by R6, the *Nominators* actor is related with the *Collect Completed Forms* use case. The explanation for the other relationships is similar.

3.3 Getting Use Case Descriptions

This subsection addresses the generation of use case descriptions from a private business process model. We define a template to represent a use case description based on a simplification of the template presented by Cockburn in [12]. The proposed template is composed by six fields, which are named and described in Table 1.

Cockburn says that a real big and complex system can be modeled with only seven use cases [12]. This yields very complex use cases with several alternative scenarios. Our approach, by transforming each BPMN activity into a different use case, yields much simpler use cases, each with a single scenario. For that reason the proposed template only attend to one (main) scenario. Pre-conditions, triggers and post-conditions enable the representation of the process flow in the use case model.

The main elements involved in a process are participants (pool and lanes), activities, gateways, events, messages, data objects, data stores and artifacts [3]. These elements are connected by connecting objects (sequence flow, message flow, associations and data associations). The approach being presented intends

Table 1. The template for describing use cases

Use Case name	The use case name identifies the goal as a short active verb phrase.
Actors	List of actors involved in the use case
Pre-Conditions	Conditions that must hold or represent things that happened before the use case starts.
Post-Conditions	Conditions that must hold at the conclusion of the use case.
Trigger	Event that starts the use case.
Scenario	Sequence of interactions describing what the system must do to move the process forward.

to transform business process elements, and their associated information, in a controlled set of sentences in NL, following the CREWS guidelines.

The activity name is the use case name in the use case template. The related pools or lanes represent the actors related with the use case in the use case template, as described in sub-section 3.1.

Focusing our attention on a use case, all incoming connections and outgoing message flows, data associations, and sequence flows to events of the corresponding activity must be reflected in the use case descriptions, fulfilling the use case template previously defined.

Sequence flows outgoing an activity to a gateway or to another activity do not create a sentence in the source activity description because these connections already create sentences in the activity that receives the sequence flow.


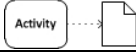

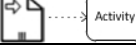
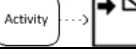



Each connecting object makes a connection between a source (sourceRef) and a target (targetRef). Different connecting objects connect different elements. The next sub-sections describe how incoming and outgoing connections of an activity are represented in the corresponding use case template.

Data Associations. Data associations are used to move data between data objects (or data stores) and activities [3]. The data (physical document or information) that are created, manipulated, and used during the execution of a process are represented as data objects (or data object references) or as data stores (or data store references). A data object reference is a way to reuse data objects in the same diagram [3]. The same happens with the data store reference.

The sentences generated by data associations and associated data objects, or data stores, are represented in Table 2. The sentences will be appended to the scenario of the use case description of the use case that represents the activity.

Association. An association is used to link text annotations and other artifacts with other BPMN graphical elements [3]. When an association links a text annotation with an activity, the text is transcribed to the scenario of the use case that

Table 2. The use case sentences originated by Data Associations

Data	Graphical representation	Originated sentence in use case scenario.
Data Object as data association source		Receives <data object name>.
Data Object as data association target		Sends <data object name>.
Data Input		Receives <data object name>.
Data Input Collection (Input set)		Receives a collection of <data object name>.
Data Output		Sends <data object name>.
Data Output Collection (Output set)		Sends a collection of <data object name>.
Data Store as data association source		Reads information from <data store name>
Data Store as data association target		Writes information on <data store name>

represents the activity. The text remains the same. When an association links a text annotation to a gateway, or to a sequence flow, the text is transcribed to the scenario of the use case that represents the target activity.

Message Flow. A message flow connects two pools representing the message exchange between the two participants [3]. A message represents the content of a communication between two Participants [3]. A Message is graphically represented as an envelope as we saw in Figure 1. The sentences originated by a message flow are described next as two different rules (MR1 and MR2).

- MR1: When an activity receives a message (message input), the use case that represents the activity will have the following sentence in its use case scenario: **Receives <message name> [with <messageRef>] from <participant name>.**
- MR2: When an activity sends a message (message output), the use case that represents the activity will have the following sentence in its use case scenario: **Sends <message name> [with <messageRef>] to <participant name>.**

MessageRef defines the message that is passed via message flow. It can be any kind of information exchanged between different pools (an email, a phone call, a document, etc.).


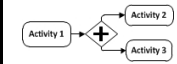
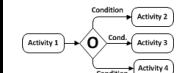

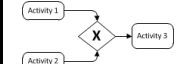
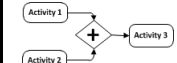


Sequence Flow. A sequence flow is used to show the order that activities are performed in a process [3]. A sequence flow can connect activities, events and gateways [3]. When a sequence flow connects two activities, it originates the next sentence as pre-condition in the use case that represents the target activity: **The <source activity name> has been completed.**

Everything that occurs between two activities must be registered in the target activity description. Involved gateways and events are treated in the next sub-sections.

Sequence Flow and Gateways. Gateways are used to control how the process flows, by diverging (splitting gateways) and converging (merging gateways) sequence flows. Splitting gateways have one incoming sequence flow and two or more outgoing sequence flows. Merging gateways have two or more incoming sequence flows and one outgoing sequence flow [3], as we can see in Table 3.

The gateway’s outgoing sequence flows may have a *Condition* that allows to select alternative paths. Each outgoing sequence flow originates a sentence represented as a pre-condition in the use case description of the sequence flow target activity. The generated sentences are represented in Table 3.

Table 3. The use case pre-condition originated by gateways

Gateway	Graphical representation	Originated Pre-condition.
Exclusive Decision		The <gateway condition> is <sequence flow condition>.
Parallel splitting		The <source name> has been completed.
Inclusive Splitting		The <sequence flow condition> is true.
Complex Splitting		The <sequence flow condition> is true.
Exclusive merging		The <source name> [exclusive or <source2 name>] has been completed.
Parallel join		The <source name> [and < source2 name>] has been completed.
Inclusive merging		The <source name> [or <source2 name>] has been completed.
Complex merging		The <source name> [or <source2 name>] has been completed.

Sequence Flow and Events. An event is something that happens during the course of a process and that affects the process's flow [3]. These events usually have a cause or produce an impact [3]. In BPMN 2.0 there is a large number of event types, so we present a general overview of the generic sentences originated in the use case template by the different events categories (see Table 4). Each category has its own table to address differences that can exist between sentences generated by the events of the same category. Due to lack of space only the sentences generated by catching events are presented here (Table 5).

Table 4. Generic sentences originated by events

Event type category	Generic sentence originated in use case template
Start	Trigger: The <event name - event definition> occurred.
Intermediate Catching	Trigger: The <event name - event definition> is received.
Intermediate Boundary Interrupting	Scenario: If the <event name - event definition> occurs, the <activity name> is interrupted.
Intermediate Boundary Non-Interrupting	Scenario: The <event name - event definition> occurred.
Intermediate Throwing	Post-condition: The <event name - event definition> is created.
End	Post-condition: The <event name - event definition> is created. The process ends.

The events affect the sequence or the timing of the process's activities. There are three types of events: Start, Intermediate and End. Start events indicate where a process (or a sub-process) will start. End events indicate where a path of a process will end. Intermediate events indicate where something happens somewhere between the start and end of a process [3].

Some events are prepared to catch triggers. These events are classified as catching events. Events that throw a result are classified as throwing events. [3]. All start events and some intermediate events are catching events[3]. The sentence originated by a catching event is included as a trigger in the description of the use case that represents the activity that is started by the event. Catching events are represented as triggers because this events cause the start of the activity.

All end events and some intermediate events are throwing events [3]. The sentences originated by the throwing events are included as a post-condition in the description of the use case that represents the activity that throws the event. Throwing events are represented as a post-condition because the event is a consequence (or a result) of the activity execution.

Table 5. The sentences originated by catching events

Catching Event	Originated sentence in use case trigger .
None	The event <event definiton> occurs.
Message	The message <event definition> arrives from <source>.
Timer	The time-date <event definition> is reached.
Conditional	The condition <expression> become true.
Signal	The signal <event definition> arrives.
Multiple	The <event definition> [or <event definition>] occurs.
Parallel Multiple	The <event definition> [and <event definition>] occurs.

Some events can also be classified as interrupting or non-interrupting events. Interrupting events stop its containing process whenever the event occurs. When Non-Interrupting events occur its containing process is not interrupted [3].

An event can be thrown by an activity and caught by another. In this case the event originates a sentence in the post-condition of the use case representing the activity that throws the event and another sentence in the trigger of the use case representing the activity that catches the event.

In the next subsection the defined approach is applied to the Nobel Prize example.

3.4 Nobel Prize Example

For reasons of space, we cannot show the complete example here. So, we select the use cases that cover a greater number of application cases.

As we can see in Figure 1, the *Send Nomination Form* activity has four incoming connections: a sequence flow from an event, giving origin to a sentence in use case trigger (Table 5), an incoming message flow, a data association and an association, each one generating a sentence in use case scenario. The corresponding use case descriptions are presented in Table 6.

The *Send List of Preliminary Candidates* activity has two incoming connections: a sequence flow from a gateway, giving origin to a pre-condition (Table 3) and a data association giving origin to a sentence in use case scenario. The activity also has an outgoing message flow to *Expert* participant generating a sentence in use case scenario (Table 2). The corresponding use case descriptions are presented in Table 7.

The *Write recommendations* activity has an incoming sequence flow from a gateway, giving origin to a pre-condition (Table 3) and an outgoing data

Table 6. Send Nomination Form use case description

Use Case name	Send Nomination Form.
Actors	Nobel Committee, Nominator
Trigger	The time-date September is reached.
Scenario	Around 3000 invitations confidential nomination forms are sent to selected Nominators. Reads information from Nominators. Sends the Nomination Invitation to Nominator.

Table 7. Send List of Preliminary Candidates use case description

Use Case name	Send List of Preliminary Candidates.
Actors	Nobel Committee, Expert
Pre-condition	The Expert Assistance Required? is Yes.
Scenario	Reads information from Preliminary Candidates. Sends the List of Candidates to be Assessed to Expert.

Table 8. Write Recommendations use case description

Use Case name	Write Recommendations.
Actors	Nobel Committee
Pre-condition	The Expert Assistance Required? is No or Select Final Candidates has been completed.
Scenario	Sends The Report with Recommendations.

association giving origin to a sentence in the use case scenario. The corresponding use case descriptions are presented in Table 8.

4 Conclusions and Future Work

This paper presents an approach to generate a use case model, including descriptions, from a private BPMN process diagram. The approach starts by presenting a set of rules to generate the 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. To identify the use cases description a set of structured sentences

are created in NL. Each sentence represents an incoming or outgoing connection from the use case corresponding activity.

BPMN has originally been design to be a language easy to understand by all stakeholders involved [27,3], nevertheless with the increase in number of its graphical elements, in its most recent version (BPMN2.0), the language has become more complex and consequently difficult to understand. The approach presented herein helps understanding BPMN models as it translates a model to NL, promoting the understanding between the involved stakeholders.

The BPMN2.0 allows business process models to be highly detailed. This is good news if one intends to use BPMN models as a basis to the development of the software that supports the business. The presented approach benefits from a detailed business process model, as greater business process detail yields a more complete use case model.

Generating a complete use case model from a business process model 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 [28]. The presented approach enables traceability between business processes and the corresponding elements in software models.

Typically, in a real situation, a software product does not support only one business process, but rather a set of processes. So, in order to generate a complete use case model for the development of such software product, we intend to extend the approach presented herein to generate a use case model representing the set of processes that comprise a business.

When sub-processes are involved, this approach demands that they are fully expanded, losing some structuring information. As future work, we intend to treat the sub-processes by refining the use cases in different detail levels.

References

1. van der Aalst, W.M.P.: Business process management demystified: A tutorial on models, systems and standards for workflow management. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) ACPN 2003. LNCS, vol. 3098, pp. 1–65. Springer, Heidelberg (2004)
2. Ko, R.K.L.: A computer scientist’s introductory guide to business process management (bpm). *Crossroads* 15, 11–18 (2009)
3. OMG, Business process model and notation (BPMN), version 2.0. Tech. rep., Object Management Group (2011)
4. Jalote, P.: *A concise Introduction to Software Engineering*. Springer (2008)
5. Mili, H., Jaoude, G.B., Lefebvre, É., Tremblay, G., Petrenko, A.: Business process modeling languages: Sorting through the alphabet soup. In: OOF 22 NO. IST-FP6-508794 (PROTOCURE II) (September 2003)
6. Shishkov, B., Xie, Z., Liu, K., Dietz, J.L.: Using norm analysis to derive use cases from business processes. In: Proc. 5th Workshop on Organiz. Semiotics (2002)
7. Hull, E., Jackson, K., Dick, J.: *Requirements Engineering*. Springer (2011)

8. Dietz, J.L.G.: Deriving use cases from business process models. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 131–143. Springer, Heidelberg (2003)
9. Bittner, K., Spence, I.: Applying use cases: a practical guide. P. Ed. inc. (2003)
10. Roussev, B.: Generating OCL specifications and class diagrams from use cases: a newtonian approach. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, p. 10 (January 2003)
11. Fantechi, A., Gnesi, S., Lami, G., Maccari, A.: Applications of linguistic techniques for use case analysis. *Req. Eng.* 8(3), 161–170 (2003)
12. Cockburn, A.: *Writing Effective Use Cases*. Addison Wesley (2001)
13. Meyer, A.: Data in business process modeling. In: Proceedings of the 5th PhD Retreat of the HPI Research School on Service-oriented Systems Engineering (2010)
14. Booch, G., Rumbaugh, J., Jacobson, I.: *The Unified Modeling Language User Guide*. Addison Wesley (1998)
15. OMG, Unified modeling language (OMG UML), version 2.5. Tech. Rep., Object Management Group (2012)
16. Giaglis, G.M.: A taxonomy of business process modeling and information systems modeling techniques. *International Journal of Flexible Manufacturing Systems* 13, 209–228 (2001)
17. Dijkman, R.M., Joosten, S.M.: Deriving use case diagrams from business process models. Tech. rep., CTIT Tech. Rep., Enschede, The Netherlands (2002)
18. Dijkman, R.M., Joosten, S.M.: An algorithm to derive use cases from business processes. In: 6th Int. Conf. on Software Engineering and Applications (2002)
19. Rodríguez, A., Fernández-Medina, E., Piattini, M.: Towards obtaining analysis-level class and use case diagrams from business process models. In: Song, I.-Y., et al. (eds.) ER 2008 Workshops. LNCS, vol. 5232, pp. 103–112. Springer, Heidelberg (2008)
20. Rodríguez, A., Fernández-Medina, E., Piattini, M.: Towards CIM to PIM transformation: From secure business processes defined in BPMN to use-cases. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 408–415. Springer, Heidelberg (2007)
21. Bittner, K., Spence, I.: *Use Case Modeling*. Pearson Education Inc. (2003)
22. Rolland, C., Achour, C.B.: Guiding the construction of textual use case specifications. *Data & Knowledge Engineering* 25, 125–160 (1998)
23. Cox, K.: Heuristics for use case descriptions. Thesis (PhD) (November 2002)
24. Phalp, K., Vincent, J., Cox, K.: Improving the quality of use case descriptions: empirical assessment of writing guidelines. *Software Quality Journal* 15(4), 383–399 (2007)
25. Allweyer, T.: *BPMN 2.0 - Introduction to the standard for business process Modeling*. Books on Demand GmbH, Norderstedt (2010)
26. OMG, BPMN 2.0 by example. Tech. Rep., Object Management Group (2010)
27. Magnani, M., Montesi, D.: BPDMMN: A conservative extension of BPMN with enhanced data representation capabilities. In: CoRR (2009)
28. Santos, M.Y., Machado, R.J.: On the derivation of class diagrams from use cases and logical software architectures. In: 2010 Fifth ICSEA (2010)