Visualizing Use Case Sets as BPMN Processes

Daniel Lübke and Kurt Schneider
Leibniz Universität Hannover, FG Software Engineering
{Daniel.Luebke,Kurt.Schneider}@inf.uni-hannover.de

Matthias Weidlich
Hasso Plattner Institute, Potsdam
Matthias.Weidlich@hpi.uni-potsdam.de

Abstract

The goal of many software projects is the support of business processes. A typical business process spans multiple Use Cases. This poses the difficulty of making the set of encompassing Use Cases consistent with each other and functionality-wise complete with regards to the overall business process. Manual arrangements and reviews of Use Cases are burdensome and time-intensive. Therefore, an automatic approach is needed that restores an overview of the Use Cases and visualizes the control-flow of the resulting business process. Our approach uses pre-conditions, post-conditions, and triggers of Use Cases to automatically assemble business processes in BPMN notation in order to solve this problem.

1. Introduction

Software projects have become larger over the time and therefore the software requirements specifications have grown as well. One way to document requirements is the use of Use Cases, which describe the interaction of an actor with the desired system. However, as the number of Use Cases increases with the described functionality, the overview, the dependencies and possibly the execution order of Use Cases is lost more and more. Therefore, the task of locating specific Use Cases in large documents and avoiding contradictions becomes a hassle. Within this paper we describe the generation of graphical business process models in BPMN notation as a mean to restore the global overview for the Use Cases and to order them according to their pre-conditions and post-conditions. During requirements elicitation, visualization can make requirements engineers and stakeholders aware of wrong conditions and Use Case specifications.

This paper is structured as follows: In the next section we describe the related work that deals with scenarios and Use Cases and their graphical representation. In the third section, tabular Use Cases and BPMN are introduced shortly. Afterwards, the generation algorithm of BPMN processes from Use Case sets is described. This constitutes the main contribution of this paper. In section five we demonstrate our approach with a short example before we conclude and present a short outlook.

2. Related Work

There is some related work concerned with the relationship between (graphical) business processes and Use Cases and generation between the two model types.

In his book about Use Cases, Cockburn mentions the possibility of applying Use Cases for deriving business processes. He offers a template in [1] but no rules or advice on how to proceed from there.

The field of model-driven development has tried to integrate the concept of Use Cases within its UML models. Instead of tabular and textual descriptions, UML sequence diagrams or similar models are used [4]. A Use Case is denoted in Use Case Diagrams and its scenario is refined in other UML models. Thereby the textual description is eliminated. This can pose a problem when communicating with non-technical stakeholders because they are not used to (technical) graphical notations. A process for UML-based development of business processes is given in [6]. From such approaches capabilities for expressing control-flow between Use Cases are missing and as such is the generation of global business process models.

An existing approach for business process generation in UML is to model the control-flow between Use Cases explicitly in at least one additional model. With the introduction of Use Case Charts and their formalization [9], it is possible to define control-flow dependencies between Use Cases and to refine them further in UML. In this approach

Table 1. Example Use Case in tabular Form

Use Case No. 1	Student applies for Thesis
Preconditions	(none)
Trigger	Student wants to write Thesis
Postconditions	Application is submitted
Main Scenario	1. Students fills out form with personal data 2. System acknowledges receipt of data on screen and by e-mail
Extensions	2a If form is not filled out completely goto step 1.

Use Cases are called scenarios that may not have extensions and that are modelled as UML sequence diagrams. However, dependencies between Use Cases cannot be derived from the Use Case themselves but have to be modeled explicitly.

An approach by Somé for synthesising state transition graphs from Use Cases is better addressing the visualization aspect [7]. The metamodel used by Somé relates to tabular Use Case descriptions. However, it is not as powerful as Use Cases defined here because extensions must not have further extensions. A tabular Use Case can be converted to a state transition graph similar to graphical business process languages. However, Somé's approach does not deal with combining many Use Cases to a larger state transition graph. Thus, the state transition graphs can be used for simulating one Use Case [8] but are not suited for visualizing dependencies between Use Cases.

A predecessor to our work here is the generation of EPC models from Use Cases [3]. EPC models consist of fewer graphical symbol types but are not as powerful as BPMN. For example, BPMN can differentiate between different event types (plain event, time-based event and so on). Furthermore, BPMN seems to become *the* standard business process modeling language. Therefore, the transformation of Use Cases should have BPMN as the target notation.

3. Use Cases and BPMN

Use Cases are an established technique for capturing and documenting requirements for software systems. Use Cases can be captured in different ways [1]. Within this paper we assume Use Cases to be in a tabular, semi-structured form. Use Cases describe the interaction of actors (usually the users) with the desired system. An excerpt with the relevant parts of a tabular Use Case can be seen in table 1.

The main part of a Use Case is its main scenario in which actors perform activities and the system responds accordingly. The main scenario describes the success case of the Use Case. Exceptions and non-success cases can be added with extensions. An extension is a new scenario that can be linked to steps and can be performed if a certain condition is true. Furthermore, preconditions, triggers and postconditions are documented. These are the elements that are considered in our approach. Other information, like minimal guarantees or the main actor are discarded.

The Business Process Modeling Notation (BPMN) [5] as standardized by the OMG is a graphical notation for capturing business processes. It aims at documenting and communicating business processes between all stakeholders. As an easy-to-use yet powerful notation, intuitive visualization of business processes is the strength of BPMN. Therefore, it is suited for the visualization of the implicit control flow that is hidden in the Use Case descriptions.

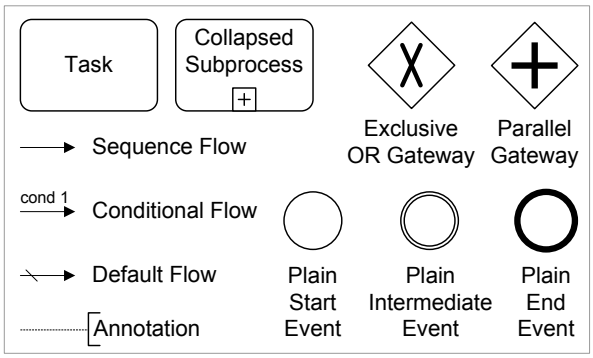

Figure 1. Subset of BPMN used in this paper

In general, BPMN is a graph-oriented approach and provides means to specify activities along with their control flow dependencies. Activities can be either tasks that represent atomic work units or subprocesses. The latter contain complete process definitions and might be collapsed for illustration purposes as shown in figure 1.

Causal dependencies between activities are expressed via sequence flow arcs. Further on, we apply a conditional flow arc in case activation is based on a condition. In order to ensure continuation of the process at a dedicated decision point, a default flow can be applied. It is activated if none of the conditional flows at the decision point are activated.

Such a decision point is modeled using an exclusive OR gateway. Whenever one of its incoming flows is activated, exactly one of its outgoing flows is activated based one the conditions of these flows. In contrast, the parallel gateway activates all of its outgoing flows, whenever all of its incoming flows are activated.

BPMN processes graphs start with a start Event and end with an end event, whereas events during the processing are modeled by intermediate events. If the type of the event

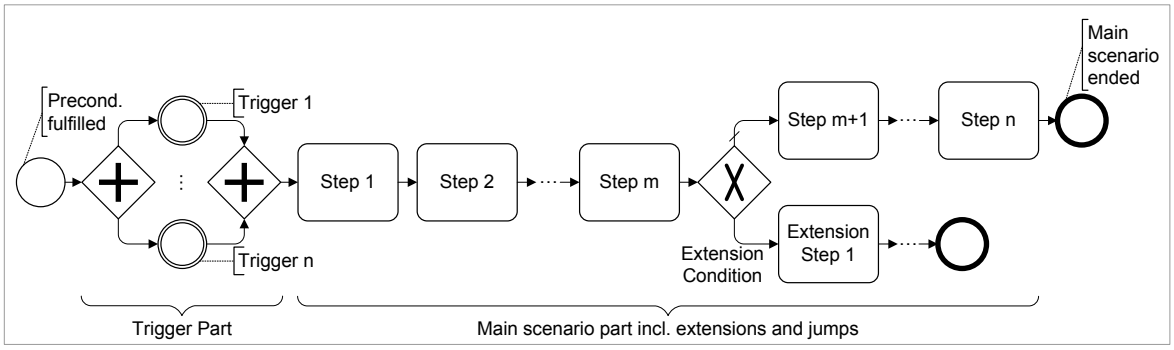

Figure 2. A BPMN process representing a single Use Case

trigger is not specified in detail, we refer to these events as plain events. However, we apply annotation elements to provide a basic description of the event trigger.

It is important to notice that the aforementioned constructs represent solely the subset of BPMN that we use to visualize Use Case dependencies. The BPMN specification introduces advanced control flow concepts (i.e. various gateway types), specialized event triggers, and means to specify the data flow and resource assignments. It is also worth mentioning that semantics of BPMN is specified informally. However, a formal definition of semantics for the subset of BPMN that is applied in this paper has been presented recently [2].

4. Generation of BPMN Processes from Use Case Sets

The following elementary steps have to be performed in order to visualize a Use Case set:

1. We have to derive a flat Use Case model. That is, all *include* and *extend* relationships of Use Cases are replaced by the scenario and the extensions of the referenced Use Case.

2. Generation of a single BPMN process for each of the Use Cases. These processes are independent of each other and consider solely the *triggers*, the *main scenario*, all *extensions*, *extensions steps*, and *jumps*.

3. All generated BPMN processes are joined based on the *preconditions* and *success guarantees* of the respective Use Case.

4. The resulting BPMN process is refactored, i.e. multiple constructs representing an identical trigger are identified such that only one of these constructs remains in the model.

As the first step comprises just trivial replacements, we focus on the algorithms realizing the other three steps in the remainder of this section. In general, a BPMN process representing a single Use Case starts with a plain start event – representing the preconditions – that is followed by one or multiple intermediate events that are executed in parallel. These events represent the triggers of the according Use Case. Further on, every step of the main scenario and potential extension scenarios is represented by a task. The causal dependencies between these tasks are captured using sequence flow connectors. Extensions and jumps are handled using exclusive OR gateways. The generation is illustrated in figure 2. Algorithm 1 defines the generation of a BPMN process for a single Use Case in pseudo code.

The main logic is contained in the conversion of scenarios which is recursive due to the recursive structure of the extensions. We illustrate the conversion in algorithm 2.

After every Use Case has been converted to a BPMN process, the processes can be joined in the third step. Therefore, the preconditions and triggers are matched. In our approach this matching is a literal one so far although more sophisticated implementations can use predicates in order to yield better results.

For each postcondition that exists as a precondition or trigger, the subprocesses representing the Use Cases are connected. Whenever two Use Cases have the same postcondition, the postconditions are joined first by introducing a gateway. If the postcondition is used by more than one Use Case, a parallel gateway is introduced to split the control flow. An example for joining four Use Cases is illustrated in figure 3. In this figure the third Use Case has only the precondition that matches the postcondition of the first and the second Use Case while the fourth Use Case has an additional trigger.

Within the fourth step of the overall generation, duplicate conditions are eliminated. In this step all events that symbolize the same condition are unified.

Algorithm 1 Creation of a BPMN process for a single Use Case

```
1:  P := new BPMNProcess();
2:  StartEvent := P.add(new
    StartEvent(UC.PreConditions));
3:  if UC.Triggers.Count > 1 then
4:     ParallelGateway := P.add(new ParallelGateway());
5:     LastElement := P.add(new ParallelGateway())
6:     StartEvent.connectTo(ParallelGateway());
7:     for all Trigger IN UC.Triggers do
8:        Event := P.add(new IntermediateEvent(Trigger));
9:        ParallelGateway.connectTo(Event);
10:       Event.connectTo(LastElement);
11:    end for
12: else
13:    LastElement := P.add(new IntermediateEvent
       (UC.Triggers[0]));
14:    StartEvent.connectTo(LastElement);
15: end if
16: ConvertScenario(UC.MainScenario, LastElement);
17: for all Step in UC.Steps do
18:    do something
19: end for
20: EndEvent := P.add(new
    EndEvent(UC.PostConditions));
21: LastElement.connectTo(EndEvent);
```

Algorithm 2 Conversion of Scenarios to BPMN

```
1:  Function ConvertScenario(Scenario, LastElement):
2:  for all Step IN Scenario.Steps do
3:     if Step.IsJumpTarget then
4:        XORGateWay := P.add(new XORGateWay());
5:        LastElement.connectTo(XORGateWay);
6:        LastElement := XORGateWay;
7:     end if
       P.add(new Activity(Step));
8:     if Step.isExtended then
9:        XORGateWay := P.add(new XORGateWay());
10:       LastElement.connectTo(XORGateWay);
11:       LastElement := XORGateWay;
12:       for all Extension IN Step.Extensions do
13:          ConvertScenario(Extension, LastElement);
14:       end for
15:    end if
16: end for
17: if Scenario.JumpsBack then
18:    LastElement.connectTo(
       GetXORGateWayFor(Scenario.JumpTarget));
19: end if
```

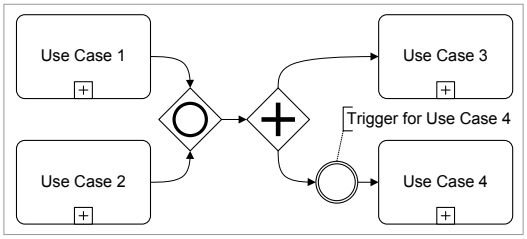

Figure 3. Example for joining BPMN Processes

5. Example

In order to illustrate the visualization of Use Cases in BPMN, we take an excerpt from a larger project. The excerpt consists of the four Use Cases located at the start of the process. The process is a university process that encompasses the management of masters' and bachelors' thesis.

Each Use Case, like *Student applies for Thesis*, have different preconditions, triggers and postconditions. The four Use Cases can be joined as illustrated in figure 4. The joining identifies that *Student applies for Thesis* can be in parallel to *Student selects Topic*. If the academic examination office has approved the application, the two branches are joined and it is the supervisors turn to approve the topic before the rest of the process can continue.

The visualization of these four Use Cases clearly denotes the possible sequences of execution and the dependencies between them. This is even more essential when dealing with more Use Cases. The original project encompassed about 60 Use Cases of which 35 were dealing with this central business process. Using the visualization it is possible to reengineer business processes and check whether preconditions, postconditions and triggers are meaningful.

6. Beyond Literal Matching of Conditions

In the above algorithms, we assume identical conditions when matching and connecting Use Cases. A match will be made only when the postcondition of a Use Case and the precondition (or guarantee) of it successor are literally the same. This assumption was made to simplify the algorithms that are the core of our contribution.

However, in many practical cases this assumption will not hold. There are a number of ways to deal with conditions that are not literally the same, but would semantically permit connection of use cases:

Logical reasoning: In principle, the string representing pre- and postconditions as well as guarantees could

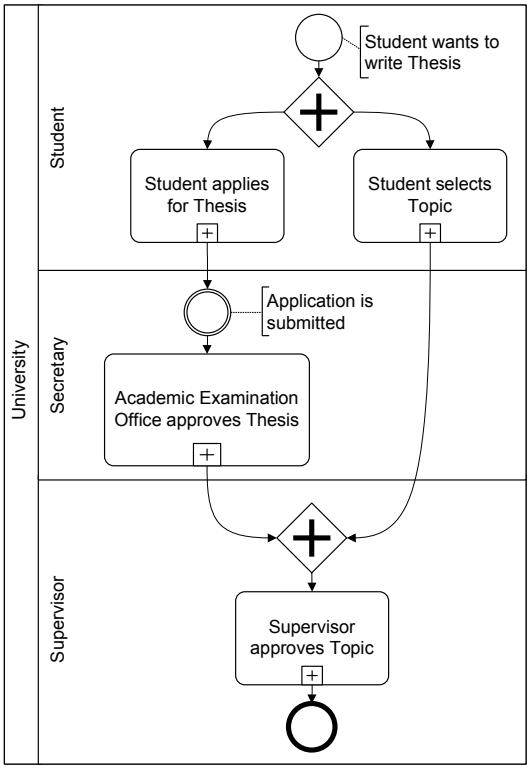

Figure 4. Example of generating BPMN from four Use Cases

be imported into a logical reasoning tool. For our approach to work, it is sufficient for a postcondition to logically imply the precondition of its successor(s). This can be integrated seamlessly into our approach and our algorithms. Our simplified algorithms assuming literal match are stricter than they need to be. This change would not make a conceptual difference to the core of our approach. Currently we have not pursued this line of work.

Human interaction using the visualizations: Human interaction using the visualizations: Use Cases represent stakeholder requirements. Those requirements can be vague or distorted by false assumptions and misunderstandings, and they can change over time. Conditions in a Use Case template may not be as precise and definite as they look. We consider it an interesting challenge to use our visualizations for confronting stakeholders. Whenever conditions do not literally match, Use Cases are not connected, according to our algorithms. However, stakeholders could be invited to interactive workshops to discuss this issue. By walking through Use Cases and business process visualizations, stakeholders may consider whether Use Cases "should be connected" from their domain perspective. Such a statement suggests: Conditions should permit connecting those Use Cases. If they do not, the *conditions* should be rephrased. This application of our visualization will extend the scope and benefit of our approach in an interesting direction in the future.

By sketching two options for dealing with conditions that do not literally match, we show possible extension of our approach. However, the core of our approach presented above is not affected by those extensions.

7. Conclusion and Outlook

Within this paper we have presented an approach to visualizing dependencies and ordering for Use Case sets. By using this approach, it is possible to create BPMN models that make the dependencies between Use Cases explicit by using preconditions, postcondition and triggers. Stakeholders and requirements engineers are therefore able to discuss the ordering and the conditions of the Use Cases, which can greatly improve the quality of the overall Use Case model.

In the future, we like to integrate case-based reasoning for better matching of the conditions as well as tool-support that is integrated into requirements engineering tools.

8. References

[1] A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley, 14th edition, August 2005.

[2] R. Dijkman, M. Dumas, and C. Ouyang. Semantics and Analysis of Business Process Models in BPMN. *Information and Software Technology (IST)*, 2008. accepted for publication.

[3] D. Lübke. Transformation of Use Cases to EPC Models. In *Proceedings of the EPK 2006 Workshop, Vienna, Austria, http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-224/*, 2006.

[4] Object Management Group. Unified Modeling Language: Superstructure. WWW: http://www.omg.org/cgi-bin/doc?formal/05-07-04, last access 2007-09-01, 2004.

[5] Object Management Group. *Business Process Modeling Notation (BPMN) 1.1*, January 2008.

[6] B. Oestereich, C. Weiss, C. Schröder, T. Weilkiens, and A. Lenhard. *Objektorientierte Geschftsprozessmodellierung mit der UML*. d.punkt Verlag, 2003.

[7] S. Somé. An approach for the synthesis of State transition graphs from Use Cases. In *Proceedings of the International Conference on Software Engineering Research and Practice*, 2003.

[8] S. Somé. Supporting Use Cases Based Requirements Simulation. In *Proceedings of the International Conference on Software Engineering and Practice (SERP'04)*, 2004.

[9] J. Whittle. A Formal Semantics of Use Case Charts. Technical Report ISE-TR-06-02, George Mason University, http://www.ise.gmu.edu/techrep, 2006.