Describing Business Processes with Use Cases

Jerzy Nawrocki
Jerzy.Nawrocki@put.poznan.pl

Tomasz Nędza
Tomasz.Nedza@sputniksoftware.com

Mirosław Ochodek
Miroslaw.Ochodek@pwsz.pila.pl

Łukasz Olek
Lukasz.Olek@cs.put.poznan.pl

Abstract

Business processes can be described with diagrams, e.g. BPMN diagrams, or as text. Use cases are a text-based notation. They are semiformal: a business process is expressed as a sequence of steps and each step is presented in a natural language. In the paper two experiments are described that aimed at comparison of diagram-based and text-based notation. Moreover, we describe some extensions to use cases which we have found interesting when working on description of business processes based on use cases. Those extensions, among others, allow to describe actor metamorphosis and specify steps which must be performed before the main scenario is executed. The ideas described in the paper have been incorporated into UC Workbench – a tool supporting editing and animation of use-case-based models.

1. Introduction

In general there are two approaches to describing business processes: diagrams and text. Many diagram notations have been proposed in Software Engineering including Petri Nets [14], Statecharts [18], HRT HOOD [2], and UML[6]. Perhaps the most popular in the area of business modeling is UML [13]. In 2004 another diagram notation has been proposed, called BPMN [3], which has potential to become an industry standard.

As regards text-based notations, from the business process description point of view the most important are use cases. They have been invented by Ivar Jacobson [8] to describe system's behavior from the end-user perspective. They are based on a natural language and in the most popular form they are presented as a sequence of steps performed by actors. Use cases have been incorporated into UML (as use-case diagrams) [6] and into the Rational Unified Process [9]. That made them very popular. That technique has been further refined by Cockburn [4], and Adolph, Bramble et al. [1]. Cockburn, Adolph, and Bramble have proposed many very useful guidelines how to write use-cases that are easy to understand.

Business process description is especially important when a computer system is to be introduced into a quite complicated business domain. Then, before starting to implement the system it can be useful to prepare a document called Concept of Operations which describes current situation, justification for change, and proposed system [7]. Current situation and proposed system can be described as text or diagrams. To check their quality they should be presented not only to IT people but also to customer representatives and prospective end-users. Thus, they should be easy to understand by a broad audience.

The aim of the paper is to present the lessons we learned while trying to select a business description notation that would be the best from the point of view of the Concept of Operations document. We have run two experiments aimed at comparing use cases with BPMN. It has turned out that it is easier to detect defects in use cases than in BPMN diagrams and even better is to augment use cases with BPMN. We have also identified some "description patterns" for use cases that can be useful from the business description point of view. Use cases as presented in the literature are oriented towards specification of human-computer interaction. At this level of abstraction the focus is on relatively short sessions taking minutes or hours. At the level of business processes, time intervals that should be considered can be as long as weeks or years. As a result one can observe new phenomena such as transition one actor into another one (actor metamorphosis).

The next two sections are short introductions to use cases and BPMN diagrams. The experiments aimed at comparison of use cases with BPMN diagrams are described in Sec. 4. The phenomena concerning actors that we have encountered when creating business process descriptions are discussed in Sec. 5. When working on business processes it is very important to understand purpose of each process and its steps. In Sec. 6 we propose use cases with prologs as a mean of expressing steps that are logically connected to a given use case but must be performed before its main scenario. The ideas described in the paper have been incorporated into UC Workbench – a tool supporting editing and animation of use-case-based models. The tool is briefly described in Sec. 7.

2. Short Introduction to Use Cases

Use cases can be presented in different format. They can be as short as one sentence describing just the purpose or they can be "fully dressed" and describe not only behaviour but also scope, precondition, trigger, priority, response time etc. (see [1]). In the context of business process description the following information about a use case seems most important:

- Use case *name* reflecting its business purpose.
- *Actors* participating in the use case (it can be a person, a system or a device).
- *Main scenario* describing step by step the most typical behaviour.
- *Extensions* to the main scenario extending a step of the main scenario with an event and corresponding alternative steps.

That form of a use case is perhaps the most popular (see e.g. [8,6,1,16]).

An example of a use case is presented in Fig. 7. The use case name is *Running a project according to PRINCE2*. It has such actors as Customer and Supplier (in Sec. 5 we will discuss the difference between main and supporting actors). The main scenario consists of 6 steps and it is accompanied by one extension.

A use case comprises a set of scenarios with the same business goal. Each extension describes another scenario. Use cases of the format presented in Fig. 7 are semiformal. They are expressed in a natural language but the description has a form of a sequence of steps. That makes it easier to understand by people who are not IT experts and allows also to automatically animate a set of use cases (see e.g. [11, 20]).

Sometimes use cases are mistaken with use case diagrams expressed in UML (a use case diagram shows use case names and actors but it neglects main scenario and extensions).

3. BPMN Diagrams in a Nut Shell

BPMN is a diagram notation designed with business modeling in mind. Its first version was published in 2004 [3, 19]. The notation is an extension of classical flow charts (it resembles UML activity diagrams). Similarly to UML sequence diagrams one can show interaction between different actors using swim lanes (if one wants to show flow of control between actors) and pools (if communication between actors is at the level of messages).

Fig. 1 contains a BPMN diagram corresponding to a use case of Fig. 7. There are three swim lanes on the diagram corresponding to Customer and Supplier, Executive and Project Manager, and Project Management Team. A subprocess corresponding to step 5 (Controlling execution of a stage) can be repeated and that is indicated on the box corresponding to the subprocess by a bended arrow. Symbol "+" indicates that a given subprocess (box) can be zoomed-in, i.e. it consists of other subprocesses or tasks (task cannot be zoomed-in).

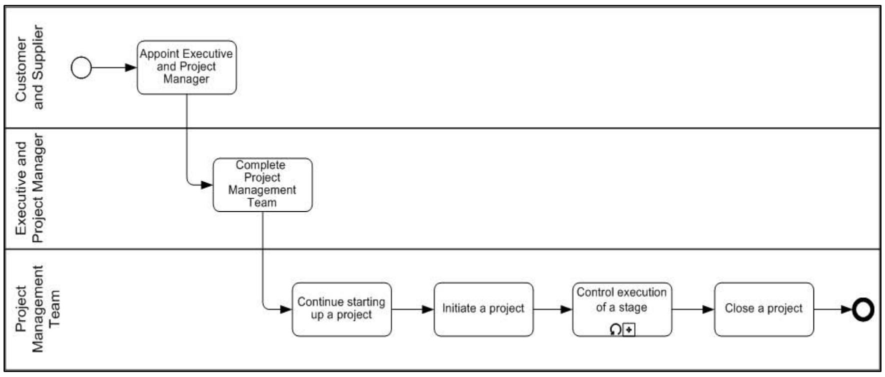

Customer and Supplier
Executive and Project Manager
Project Management Team
Appoint Executive and Project Manager
Complete Project Management Team
Continue starting up a project
Initiate a project
Control execution of a stage
Close a project

Figure 1. A BPMN diagram corresponding to the use case of Figure 7.

4. Comparison of Use Cases and BPMN Diagrams

By the two experiments described in this section we tried to answer the following questions:

- Which notation is easier to understand: use cases or BPMN diagrams?
- Does it make sense to augment use cases with BPMN diagrams?

4.1. Use cases are easier to understand than BPMN diagrams

4.1.1. Process description

The experiment was conducted in 2005 at the Poznan University of Technology. The participants were 4th year students working on their master degrees in Software Engineering (SE) or Business Administration (BA). There were 30 SE students and 11 BA students. We have split them into two groups: A and B. Group A worked with BPMN diagrams and consisted of 18 SE and 6 BA students. Group B was given description of the same business processes but expressed with use cases. It consisted of 12 SE and 5 BA students. Each group went through the following steps:

1. A *lecture* presenting an introduction to a given notation (90 minutes).
2. *Two rehearsal sessions* (each lasting for about one hour) during which the students were given a description of PRINCE2 processes [10] expressed in a given notation with a number of seeded defects and their task was to find them. The documents were 5 pages long and consisted of a short introduction, description of actors, description of business processes with use-cases (UC group) or BPMN diagrams (BPMN group), and the description of information objects (e.g. artifacts) transferred between the actors.

3. An *actual experiment session* run in a similar way as a rehearsal. Obviously, the subjects were given descriptions of new processes, but the document structure was the same.

In steps 2 and 3 the students worked individually. Every detected defect was shortly described in the defect log. As the understandability measure we have assumed the number of defects detected in the document.

There are two kinds of defects that can be found in documents and software artifacts: major and minor defects. Minor defects are less important and frequently they can be found by a skilful secretary or junior analyst. It can be wrong document structure, missing actor description, wrong spelling etc. Major defects are serious ones. It can be a logical error on a BPMN diagram, inconsistency between some use cases etc. We have assumed that understandability will be measured in terms of major defects as they require understanding of the model.

4.1.2. Results

Let defect detection ratio (DDR) for a person p and notation n ($n \in \{\text{UC}, \text{BPMN}\}$) is defined as follows:

$$\text{DDR}(p, n) = \frac{\text{Number of defects detected by person } p \text{ for notation } n}{\text{Number of all the defects in description D}(p, n)} \cdot 100\%$$

where D(p, n) denotes a description expressed with notation n given to a group of subjects containing person p (see step 3 of the experiment).

The average DDR for BPMN group (BPMN) and use-case group (UC) are presented in Fig. 2.

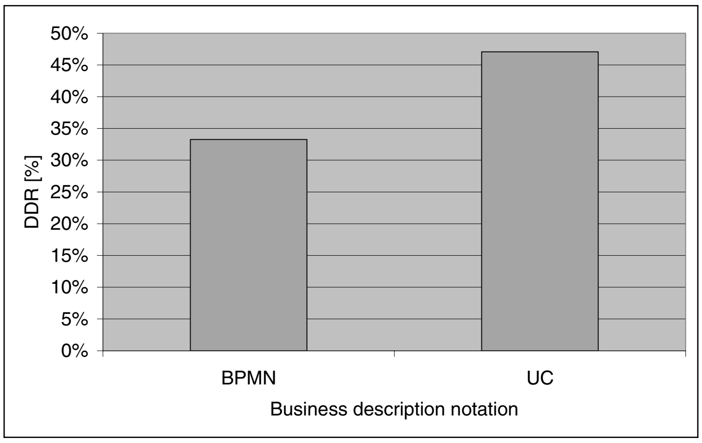


Figure 2. Average defect detection ratio (DDR) for use-case group (UC) and BPMN group (BPMN).

For each group, we have first checked that distribution of DDRs is a normal distribution (we used the Shapiro-Wilk test [17]). Then we have used the standard statistical inference procedure based on the t distribution which lead us to the following conclusion:

Average DDR for use cases is greater than for BPMN diagrams and that result is statistically significant (with the significance level 0.01). This justifies the conjecture that use cases are easier to understand than BPMN diagrams.

Thus, description of business processes should be based on use cases.

4.2. BPMN diagrams help to understand use cases

4.2.1. Process description

The experiment was conducted in 2005 at the Poznan University of Technology. We wanted to check if use cases augmented with BPMN diagrams are easier to understand than pure use cases.

The participants were the same students as in the previous experiment. In order to level the differences between students, we split them into two groups (A and B) according to their results in the first experiment. Group A consisted of 6 BA and 18 SE students, and group B consisted of 6 BA and 17 students.

Students were given a description of the Starting-up process of PRINCE2 [10]. Group A was reviewing pure use-cases, and group B was detecting defects in use-cases augmented with BPMN diagrams. They were given one hour to find the defects and put them to the defect log.

4.2.2. Results

The average DDR for each of the groups is presented in Fig. 3.

We have checked that the distribution of DDR is normal and then we have come to the following conclusion:

Average DDR for use cases augmented with BPMN diagrams is greater than for pure use cases and that result is statistically significant (with the significance level 0.01). This justifies the conjecture that use cases with BPMN diagrams are easier to understand than pure use cases.

Augmenting use-cases with BPMN diagrams can help to understand them but it takes time to develop such diagrams.

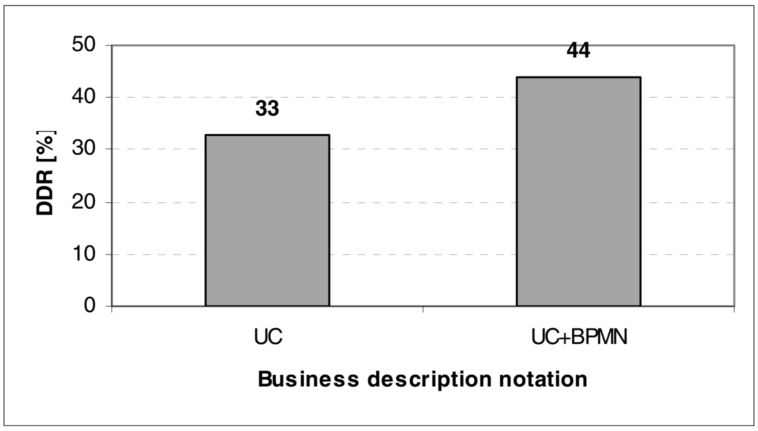


Figure 3. Average defect detection ratio (DDR) for pure use cases (UC) and use cases augmented with BPMN diagrams (UC+BPMN).

5. Actors

It is well known how to apply use cases to describe human-computer interaction (see e.g. [4,1]). However, there is an important difference between human-computer interaction (HCI) and business models. HCI processes last for minutes or hours while business processes can take weeks, months or even years (Cockburn calls the level of business processes the “summary level” [4]). When specifying business processes with use cases we have observed a different sort of phenomena concerning actors and those phenomena will be described in this section.

5.1. Main and supporting actors

Some HCI use cases contain two kinds of actors: primary and secondary. A primary actor is “an actor who wants something from the system under development at that moment” [1]. A secondary actor is usually a device or a system (e.g. a switching network).

A similar distinction can be made at the business process level. We have found it useful to have two different kinds of business actors: main actors and supporting ones. Assume someone is thinking about building a computer-based system for a dentist office. The business analyst has identified three roles: a dentist, a patient, and the dentist’s secretary. The real “business” is between the dentist and the patient – the secretary is only supporting them. Thus, the dentist and the patient would be main actors and the secretary would be supporting one. The distinction between main and supporting actors can help to understand the business and better reengineer its process before a computer system will be specified and implemented. Main actors are vital to the business and they cannot be removed by an introduction of a computer-based system, while the role of supporting actors is auxiliary one and it can be easily changed or even completely replaced by new technology.

5.2. Collective actors

At HCI level there is only one person at the moment. When specifying business models there are many situations in which a step is performed by a group of people. For instance at a university some decisions are to be made by a faculty board or by the university council.

5.3. Dynamically created actors

At the HCI level actors are static. An actor exists before a given use case is invoked and its lifetime extends at least till the end of the use case. At the business process level, since described periods of time are much longer, there are some use cases during which an actor is created.

PRINCE2 is a very popular method of project management [10]. Assume one wants to describe PRINCE2 processes with an intension of developing an internet-based application that would support project management with PRINCE2. A process map for PRINCE2 is presented in Fig. 4.

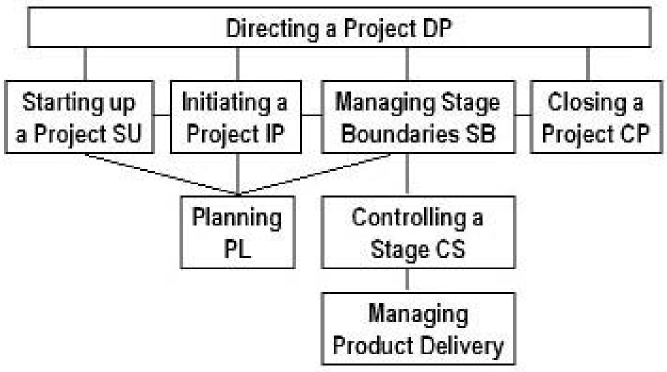


Figure 4. A process map for PRINCE2 according to [10].

The first problem is how to present this 2-dimensional description in a sequence of use-case steps. It is important to realize that modeling is strongly connected with abstraction. It is an art of neglecting unimportant details. In this case one should focus on a sequence of processes SU + IP + (SB & CS) + CP. Then the corresponding use case could be presented as in Fig. 5.

UC-1: Running a project according to PRINCE 2
Main scenario
1. Starting up a project
2. Initiating a project
3. Executing a stage
4. Closing a project

Extensions
3a. One more stage is needed.
  3a1. Step 3 is repeated.

Figure 5. Use case describing project management according to PRINCE2.

The main problem with the use case of Fig. 5 is that it does not specify who performs a given step. One could put "Someone" in the beginning of each step but it is just a syntactic trick. Another option could be the Project Management Team (PMT). But PMT is created during Starting up a Project (SU), so it cannot execute SU since at the beginning PMT does not exist. To find the answer one should zoom into the SU process which is depicted in Fig. 6.

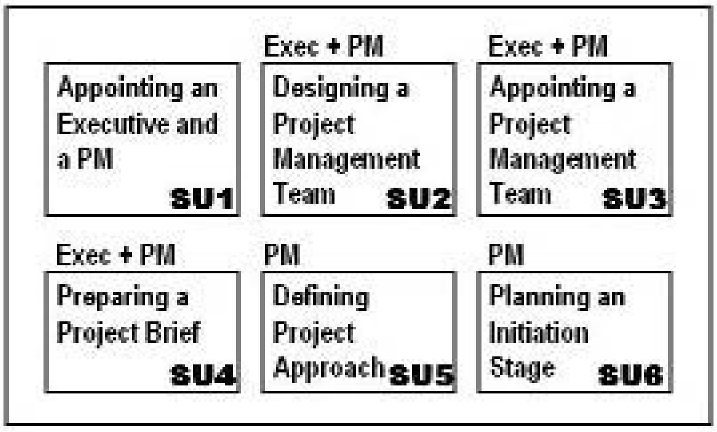

Exec + PM
Exec + PM
Appointing an Executive and a PM SU1
Designing a Project Management Team SU2
Appointing a Project Management Team SU3
Exec + PM
PM
PM
Preparing a Project Brief SU4
Defining Project Approach SU5
Planning an Initiation Stage SU6

Figure 6. Starting up a Project (SU) in PRINCE2. PM stands for Project Manager.

However, zooming into each of the PRINCE2 processes and describing each subprocess as a step (SU1, ..., SU6, IP1, ..., IP6 etc.) would result in a very long main scenario (according to Adolph et al. main scenario should contain from 3 up to 9 steps). To solve the problem one can extract subprocesses SU1, SU2, and SU3 from the SU process and put them at the level of use case UC-1 (subprocesses SU2 and SU3 have been combined in one step) as presented in Fig. 7.

UC-2: Running a project according to PRINCE 2
Main actors: Customer, Supplier
Supporting actors: Executive, Project Manager, Project Management Team
Main scenario
1. Customer and Supplier appoint Executive and Project Manager.
2. Executive and Project Manager complete Project Management Team.
3. Project Management Team continues starting up a project.
4. Project Management Team initiates a project.
5. Project Management Team controls execution of a stage.
6. Project Management Team closes a project.

Extensions
5a. One more stage is needed.
  5a1. Step 5 is repeated.

Figure 7. An improved version of a use case of Fig. 5.

5.4. Metamorphosis of actors

Assume one wants to build an information management system for the university. One of main actors will be a person willing to earn a degree at the university. At the very beginning that person is just a candidate, then she would be admitted and later on (after receiving a student book and a student ID) she would become a student. After passing all the exams and defending her bachelor thesis she would be graduated. At our university a candidate can become also a free student – it is more expensive but it is easier to get such a status (that matters when there is a big number of candidates). One can be a free student for up to one year. If you do not perform according to standards than you can be deleted from the list of students. What we propose is to introduce the alias operator to actor definition section. *X alias Y* means that the same person at a certain stage plays a role X and sometime after he becomes Y. What we need at the level of scenarioThose metamorphosis can be presented in a form of a finite automaton as in Fig. 8, and Fig. 9 contains a use case that manages the metamorphosis.

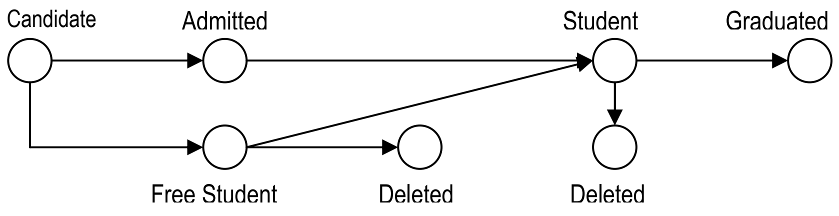


Figure 8. A finite automaton representing possible metamorphosis of a student.

UC-3: Earning a bachelor degree
Main actors: University
{Candidate **alias** Admitted **alias** Student **alias** Graduated **alias** Free Student **alias** Deleted}
Supporting actors: Senate, Rector
Main scenario
1. Senate defines the rules concerning admission.
2. Rector appoints members of the Enrolment Committee.
3. University organizes the Open Doors action.
4. Candidate makes the application.
5. Enrolment Committee announces lists of admitted. Candidate **becomes** Admitted
6. Admitted starts up his study. Admitted **becomes** a Student.
7. Student successfully passes through semesters from 1 to 7.
8. Student takes a diploma exam. Student **becomes** Graduated.
9. Graduated receives a diploma.

Extensions
5a. Candidate is not admitted due to lack of free places.
    5a1. Candidate is studying as a Free Student. Go to step 8.

Figure 9. A use case with actor metamorphosis.

6. Processes with Prologs

In business modeling processes are a central part of the description (the remaining elements are actors and information objects). To understand a business description means to know not only how its processes are executed but also for what. In Adolph's opinion "*a system is deficient if it cannot deliver services that are valuable to its users*" [1]. Paraphrasing that sentence one could say that a business process is deficient if it does not deliver services that are valuable to its main actors. Thus, a good process notation should support understanding the purpose of each process. From that point of view "classical" use cases have some weakness that are discussed below.

Assume one wants to continue development of a business model for a university, outlined in Sec. 5.4 (see use case UC-3). Assume one is going to describe step 7 (passing through a semester). The description could consist of the steps contained in the main scenario of UC-4 (see Fig. 10). However, this sequence of steps does not show activities that must be performed before a student starts a semester. We propose to extend a use case form by adding prolog to each main scenario. Prolog specifies a sequence of activities that must be executed before the main scenario. (One could think about a symmetric extension – epilog – but so far we have not found an example showing that this extension would be useful). Fig. 10 contains a use case describing how to pass through a semester and showing all the required preceding actions in a form of prolog. Prologs can increase readability of use cases by showing context for the main scenario with its extensions that can help the system analyst to focus her attention on user-valued transactions.

UC-4: Passing through a semester
Main actors: Student or Free Student
Supporting actors: Rector, Dean, Faculty Board, Director of Department
Prolog
1. Faculty Board votes a curriculum (a list of courses assigned to semesters).
2. Dean assigns courses to departments and appoints people responsible for each course.
3. At least a week before beginning of a new semester Dean announces schedule of the courses.

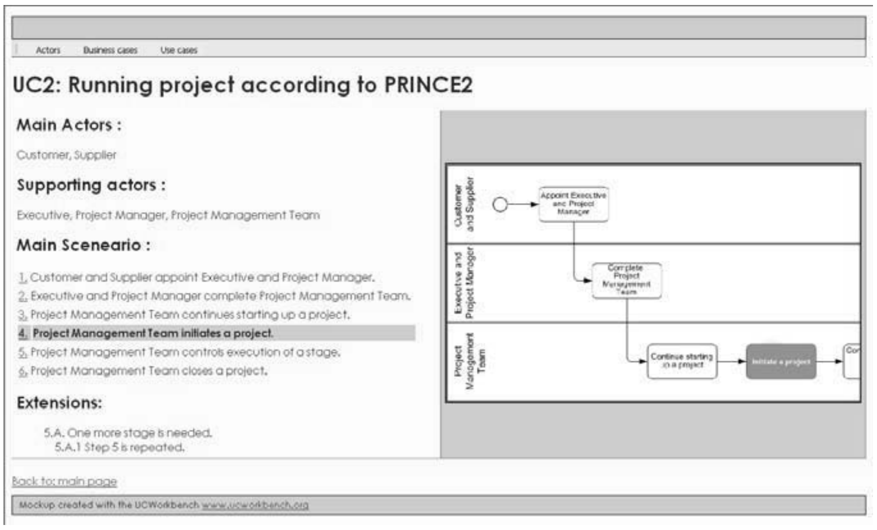
Main scenario
4 Student or Free Student attends the courses.
5 Dean sets the deadline for returning student's book for registration.
6 Student or Free Student takes exams and gets credits.
7 Student or Free Student goes for an internship.
8 Student or Free Student returns his book for registration to the Dean Office.
Extensions
7a. There is no internship assigned to this semester.
    7a1. Skip to step 8.
. . .

Figure 10. A use case with a prolog part.

7. Use Case Animation with UC Workbench

UC Workbench is a tool developed at the Poznan University of Technology [11, 20]. Originally its aim was to support use-case engineering activities at the level of human-computer interaction, i.e. editing use cases, animating them, and estimating the effort on the basis of use case points [15].

For the sake of business description we have added a few new features to the tool that aim at editing and animating business-level use cases. The previous version of UC Workbench was able to take use cases written by an analyst in the FUSE language and generate a mock-up that could animate them. At the FUSE level the analyst could "decorate" use-case steps with screen designs. To adapt UC Workbench to the needs of business modeling we decided to replace screen designs with BPMN diagrams. As a result one can animate a use case and at the same time observe the changes also on the BPMN diagram (see Fig. 11).


Actors
Business cases
Use cases
Customer and Supplier
Executive and Project Manager
Project Management Team
Appoint Executive and Project Manager
Complete Project Management Team
Continue starting in a project
Initiate a project
Back to: main page

UC2: Running project according to PRINCE2

Main Actors :

Customer, Supplier

Supporting actors :

Executive, Project Manager, Project Management Team

Main Sceneario :

1. Customer and Supplier appoint Executive and Project Manager.
2. Executive and Project Manager complete Project Management Team.
3. Project Management Team continues starting up a project.
4. **Project Management Team initiates a project.**
5. Project Management Team controls execution of a stage.
6. Project Management Team closes a project.

Extensions:

5.A. One more stage is needed.
5.A.1 Step 5 is repeated.

Mockup created with the UCWorkbench www.ucworkbench.org

Figure 11. A UC Workbench screen presents a use case along with a BPMN diagram.

Another ability of UC Workbench is generating a paper version of the model on the basis of description expressed in the FUSE language. The generated paper version (document) consists of the following sections:

1 Introduction
2 Actors
3 Business processes
4 Information objects

The main advantage is consistency: if you change something in the FUSE description that change will be immediately depicted in the mock-up and in the document.

8. Conclusions

In the paper we have discussed some problems concerning business process description with use cases. The experiments described in Sec. 4 have shown that it is easier to understand use cases than BPMN diagrams (the defect detection ratio for use cases was greater than for BPMN diagrams and this is statistically significant). Thus, it is reasonable to assume use cases as a basis for describing business processes (BPMN or UML diagrams can be a valuable add-on).

We have also encountered a number of phenomena concerning actors. Since time scale of business processes is much greater than that of human-computer interaction sessions, one can observe dynamically created actors and actor metamorphosis (see Sec. 5).

Another description practice we have learned is extending main scenario of a use case with a prolog specifying the steps that must be performed before the main scenario can be executed.

9. Acknowledgements

First of all we would like to thank the students participating in our experiments. We owe also a lot to the software companies belonging to the XPrince Consortium. Special thanks go to Piotr Godek and Grzegorz Leopold from PB Polsoft Ltd – they provided us a valuable feedback on the ideas presented in the paper.

This work has been financially supported by the Poznan University of Technology under a research grant BW 91-429.

10. References

1. Adolph S., Bramble P., Cockburn A., Pols A.: Patterns for Effective Use Cases. Addison-Wesley (2002).
2. Burns, A., Wellings, A. J.: HRT-HOOD: a structured design method for hard real-time systems, Real-Time Systems, Volume 6, Issue 1, p.73-114, January 1994.
3. Business Process Modeling Notation Version 1.0, May 2004 (http://www.bpmn.org).
4. Cockburn A.: Writing Effective Use Cases. Addison-Wesley (2000).
5. Douglas C.: Introduction to Statistical Quality Control, Third Edition. Wiley & Sons (1997).
6. Fowler M., Scott K.: UML Distilled. Addison-Wesley (2000).
7. IEEE Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document, IEEE Std 1362-1998, March 1998.
8. Jacobson I., Christerson M., Jonsson P., Overgaard G.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, Reading MA (1992).
9. Kruchten, P.: The Rational Unified Process: An Introduction (2nd Edition). Addison-Wesley (2000).
10. Managing Successful Projects with PRINCE2. The Stationery Office Books (2002).
11. Nawrocki J., Olek Ł.: A Tool for Use-Case Engineering. Extreme Programming and Agile Methodologies 2005, Lecture Notes in Computer Science 3556, 230-234.
12. OMG Unified Modeling Language Specification Version 1.5, March 2003 (http://www.omg.org/technology/documents/formal/uml.htm).
13. Penker M., Eriksson H. E.: Business Modeling With UML: Business Patterns at Work. Addison-Wesley (2000).
14. Reisig W.: Petri Nets, An Introduction. EATCS, Monographs on Theoretical Computer Science, W.Brauer, G. Rozenberg, A. Salomaa (Eds.), Springer Verlag, Berlin (1985).
15. Ribu K.: Estimating Object-Oriented Software Projects With Use Cases. Master of Science Thesis. University of Oslo (2001).
16. Schneider G., Winters J. P.: Applying Use Cases: A Practical Guide. Addison-Wesley (1998).
17. Shapiro-Wilk test: http://www.itl.nist.gov/div898/handbook/prc/section2/prc213.htm

18. Harel D.: Statecharts: A visual formalism for complex systems. Weizmann Institute of Science, Dept. of Computer Science (1986).
19. White S.: BPMN and Business Process Management: http://www.bpmn.org/Documents/6AD5D16960.BPMN_and_BPM.pdf
20. Nawrocki J. Olek L.: Use Cases Engineering with UC Workbench. in Zielinski K., Szmuc T. (eds): Software Engineering: Evolution and Emerging Technologies, IOS Press 2005, 319-329.