The Transition from Business Modeling to Requirements Specification

Svatopluk Štolfa

Dept. of Computer Science, FEEI, VŠB-TU Ostrava, 17.listopadu 15, Ostrava-Poruba,
svatopluk.stolfa@vsb.cz

Abstract. Business modeling is the first step of software development. It is also the first step that we leave out. This paper describes how to use business process, captured by UML activity diagrams, for creating use case specification.

1. Introduction

Business modeling is the initial phase of most software processes. Next phases are use case modeling, analysis, design, etc. Use case modeling and other phases of software processes are very well defined now. Transitions between these phases are formally defined and clear. However, transition between business modeling and use case modeling is obscure. That is the reason why we often skip this phase. We don't know how a business process may be useful for software development. We don't want to waste job time producing something ineffective and useless for us. We gather requirements by creating use case model directly. The goal of our work is to define formal methodology and usable technique for transition between business model and other phases of software process. It means that this newly developed technique should be used for software development by using business modeling for specification of next phases of software process.

Initial phases of software process and their usage are:

Business process is a set of one or more linked procedures or activities, which collectively realize a business objective or policy goal. Business process may be applied in two ways. We can use business process as a goal or as a tool. We can take advantage of both approaches during the development of software. Business process gives us an opportunity to manage project effectively by organization, simulation and realization of accurate planned processes. On the other hand, we can benefit from using business process for purposes of use case specification. From this point of view, the main goal of the business process modeling is to provide common language for communities of software and business engineers. Unfortunately, this very important aspect of usage of business processes is not sufficiently covered by current technologies.

The goal of the **requirements** workflow is to describe what the system should do by specifying its functionality. Requirements modeling allows to the developer and the customer to agree with that description. Use case model examines the system functionality from the perspective of actors and use cases. An actor is someone (user) or

something (other system) that must interact with the system being developed. A use case is a pattern of behaviour the system exhibits. Each use case is a sequence of related transactions performed by the actor and the system in a dialog. We describe the use case model by UML use case diagrams. We can structure use cases by "include", "extends" and "generalization" relationships and actors by "generalization" relationship.

We can present actual results of our research that is focused mainly on business modeling. The next logical step is to use methods originally applied for business modeling for purposes of the requirements specification. This research deals with methods for transition between business process captured by UML activity diagram and use case model captured by UML use case diagram.

The transition from business modeling to requirements specification is not formally defined yet, but we can use our method to support intuition. Our method comprises the following steps. First, a business process is modeled. From the business process, use cases are derived. Afterward we structure use cases by applying *mapping patterns*.

2. Mapping Activities to Use Cases

We describe business processes by UML activity diagrams. We can capture states of business processes by activities and describe transitions between these states. The purpose of activity diagrams is to describe control flow given by inner mechanism of the execution.

We will present our method using the following example. Our example deals with an information system of video library. At first, we should determine what activities will be provided by that system. Remaining activities are enacted by humans. We exclude human activities, but we can leave their symbols in the diagram for purposes of better understandability.

Activities comprise both use cases and their components. We must decide which of activities are the parts of use cases and which of them compose the use case as the whole. We have two main methods how to map activities to use cases. These two methods are "mapping several activities to one use case", "mapping one atomic activity to one use case.

2.1 Mapping Several Activities to One Use Case

This method is the first step of above mentioned mapping (see Fig. 1). We decide that three activities (activities provided by our system) compose one use case (Lending) on the left side. "Lending" provides all system interactions with actors related to the lending process. Two activities remain on the right side. These activities compose the use case called "Returning". "Returning" provides system interactions related to returning process.

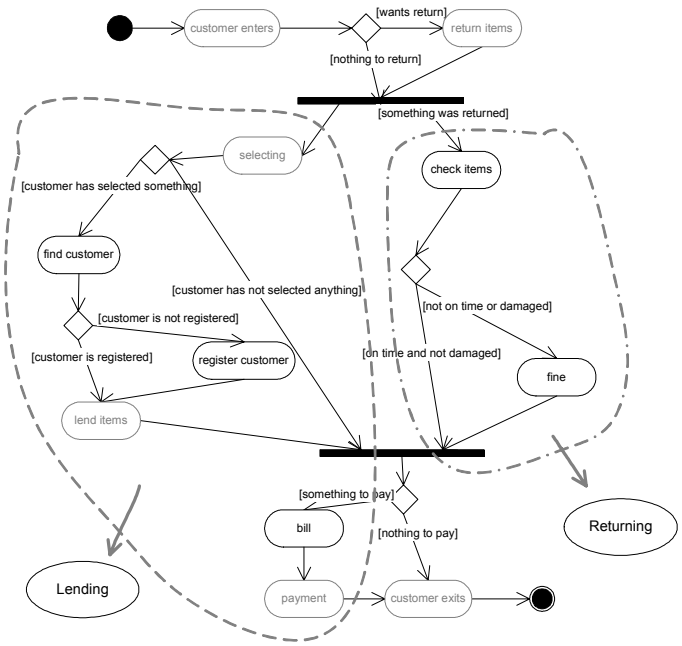

Fig. 1. Mapping several activities to one use case.

2.2 Mapping One Atomic Activity to One Use Case

We can map atomic activities (simple activities) to simple use cases by this method. Usually, it's called one-to-one mapping (see Fig. 2). Our example contains five simple activities: "find customer", "register customer", "bill", "check items" and "fine". Use cases, created from these activities, have the same or slightly modified names for better transparency.

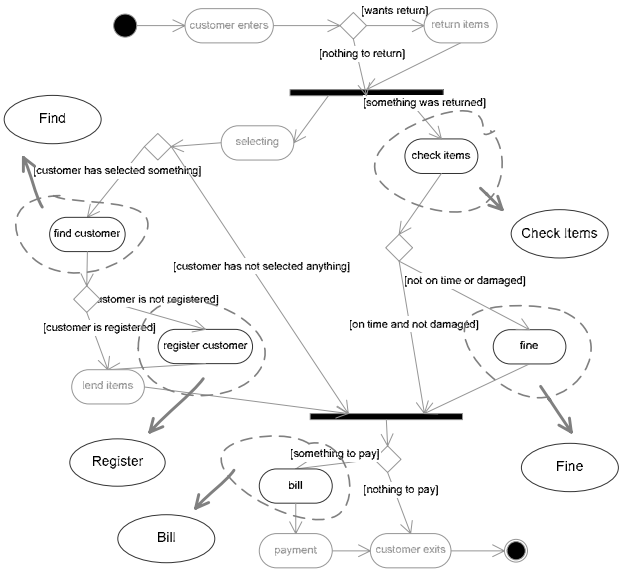

Fig. 2. Mapping one atomic activity to one use case.

3. Structuring Use Cases by Patterns

Now, we have several use cases. But these use cases are not connected to each other. UML use case diagram allows structuring use cases by relationships “extends”, “include” and “generalization”. We developed several basic mapping patterns to structure use cases according to the context of the original activities. We will present two of them. We called these two patterns “Sequential Pattern” and “Optional Pattern”.

3.1 Sequential Pattern

Intent: Organize use cases derived from sequential activities by “include” relationships.

Name of this pattern is derived from sequential execution of activities. Several activities are mapped to one use case using “mapping several activities to one use case”. Sequential activities are mapped to use cases by one-to-one method (see Fig. 3). Pattern uses “include” relationship to connect two use cases, derived from sequential activities, to the main use case. The “include” relationship indicates that “UseCase1” uses (includes) “UseCase Activity2” and “UseCase Activity3”.

This pattern should be used for mapping of a set of sequential activities.

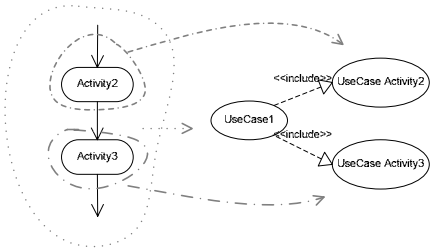

Fig. 3. Sequential pattern.

3.2 Optional Pattern

Intent: Organize use cases derived from optional activities by "extends" relationships.

This pattern shows how an optional activity can be mapped and structured to use case diagram. The optional activity can be enacted only if condition is true. Otherwise activity is skipped (see Fig. 4). If a condition block and additional activity are parts of one use case "UseCase1", then "Activity2" should be mapped to "UseCase Activity2" which extends "UseCase1". "Extends" relationship specifies that "UseCase Activity2" may extend "UseCase1".

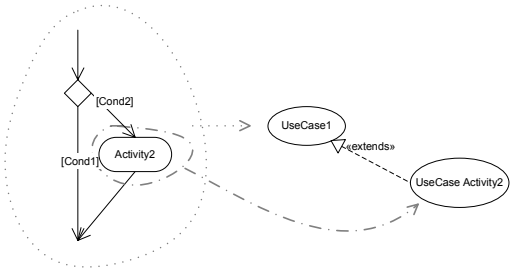

Fig. 4. Optional pattern.

This pattern should be used when optional activity is mapped to use case and all around this activity is a part of one use case.

3.3 Applying Patterns to Motivation Example

We defined two basic patterns and we can apply them now. We want to structure our use cases. It is very easy to do that by applying patterns (see Fig. 5). We used *sequential pattern* to structure use cases "Find", "Bill" and "Check Item" and *optional pattern* to organize use cases "Register" and "Fine". We excluded condition "customer has not selected anything" for better transparency, because this situation should be realized by another use case that extends "Lending". "Find" and "Register" should be organized with this excluded use case.

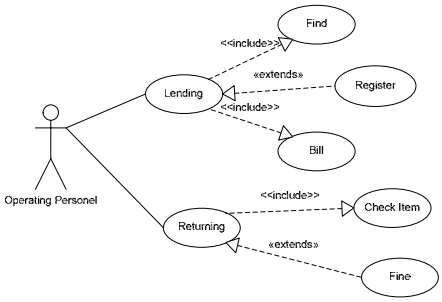

Fig. 5. Example – result.

4 Conclusions

We presented the method how the business processes can be used for use case model specification. We introduced two simple patterns and guide how to use them. However, developing use case model is more complex and this method can't solve all problems yet. Therefore, future research is focused on improvement extension and formalization of this method.

References

1. Štolfa S., Vondrák I.: „Using the Business Process for Use Case Specification", ISIM 2003, Brno, Czech Republic
2. Schneider G., Winters J. P.: "Applying Use Cases Second Edition - A Practical Guide", Addison-Wesley 2001
3. Booch, G., Rambough, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley 1998
3. Vondrák I., "Byznys Modelování", Objekty2002, Prague, Czech Republic
4. DeMarco, T.: Structured Analysis and System Specification. Englewood Cliffs, New Jersey, Prentice-Hall 1979
5. Fowler, M.: Analysis Patterns: Reusable Object Models, Reading, MA: Addison-Wesley 1995
6. Fernandez, E.B.:Building systems using analysis patterns, Procs. 3rd Int. Soft. Architecture Workshop (ISAW3), Orlando, FL , November 1998 , 37-40