UCEd: a tool for Use Cases based requirements acquisition*

Stéphane S. Somé
School of Information Technology and Engineering (SITE) University of Ottawa
800 King Edward, P.O. Box 450, Stn. A Ottawa, Ontario, K1N 6N5, Canada
ssome@site.uottawa.ca

Abstract

We present UCEd a toolset for use cases edition, use cases verification, prototype generation, and simulation. UCEd accepts use cases expressed in a constrained form of natural language. UCEd integrates the use cases in finite state machines using domain knowledge, and provides an interface for system simulation.

1. Background

Requirements engineering includes the elicitation, understanding and representation of customers needs for a system. It is a critical tasks in software engineering; the source of a great number of software failures.

The main reason for requirements induced failures is a gap existing between customers and the system development process. This gap is due to the manual nature of the requirement engineering process. Requirements are informally sought by analysts from customers who then pursue others development activities according to what they understand about customers needs. The *understanding* of requirements is generally represented as an abstract specification often not comprehensible by customers. That added to the difficulty to automatically ensure consistency between specifications and informal requirements makes difficult to ascertain, before later phases of a development process, if a specification is right according to its requirements and if there are no missing requirements.

A Use Case is "the specification of a sequence of actions, including variants, that a system (or a subsystem) can perform, interacting with actors of the system" [4]. A use case describes a piece of behavior of a system without revealing the internal structure of the system. As such use cases are effective means for requirements elicitation and analysis, and various software development approaches including the Unified Software Development Process [3] recommend use cases for requirements description.

Use cases are useful to capture and document requirements from the point of view of customers. They can also be used for prototyping and validation. The partial nature of use cases allows several customers with different views of a same system to provide different but possibly overlapping use cases describing its behavior. The partial nature also helps developing a system by incremental addition of services. A problem however, is that it is often difficult to visualize the global behavior resulting from the combination of the use cases. Moreover, separately defined use cases may be inconsistent one with the other and the set of use cases may be incomplete.

The objective of the Use Case Editor (UCEd) is to provide automatic support to use cases acquisition, use cases verification, prototype generation, and use cases simulation. UCEd takes a set of related use cases written in a restricted form of natural language and generates and executable specification that integrates the partial behaviors of the use cases [2]. UCEd uses information contained in an application domain model (a high-level class model of the system) for the syntactical analysis of use cases and specification generation.

UCEd approach is rooted in the Unified Modeling Language (UML) [4]. The domain model is a UML class diagram [5], and we assume the use cases definition and semantics of the UML specification. The UML seems appealing because of the great acceptance it has gained among software developers and tool vendors. An advantage of using UML is the possibility of integration of UCEd to the various existing UML based methodologies and tools.

2. UCEd process

UCEd is a toolset that includes a Use Case Capture module, a Domain Model Editor, a Use Case Composition Module and a Use Case Simulator. Figure 1 describes UCEd requirement engineering process involving these tools. The process starts with an early view of requirements consisting of "rough" domain model and use cases, and produces a high-level state model specification of the system as well

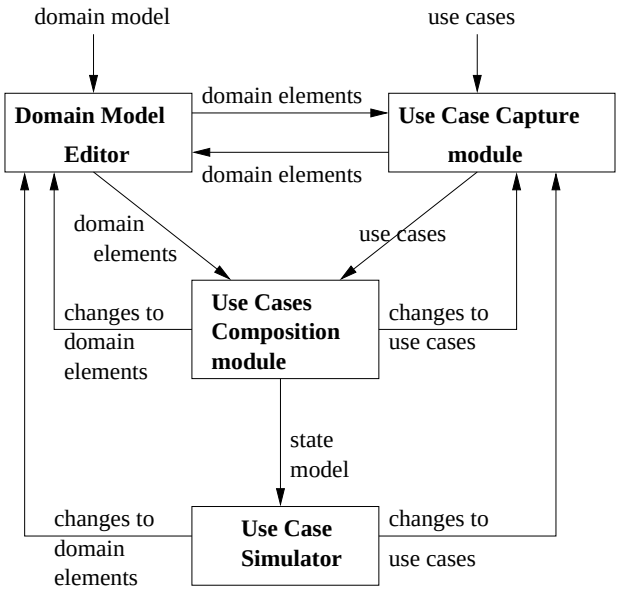

Figure 1. UCEd process. The boxes are tools and the arrows show data elements exchanged between these tools.

as clarified use cases and domain model.

A domain model is a *high-level class model* that captures *domain concepts* and their relationships. Domain concepts are the most important types of objects in the context of a system. The domain concepts include the system as a black box with the "things" that exist or events that transpire in the environment in which the system works [3]. We use UML class diagrams extended with *stereotypes* [4] to describe domain models. UCEd Domain Model Editor provides a *tree* interface for domain model capture. UCEd uses XMI [4] for interoperability with UML diagramming tools giving the possibility to load domain models created with tools such as ArgoUML[1].

The Use Case Writing module allows the edition of UML use case models and use cases contents. Use case models are described in a *tree form*. As for the domain model, UCEd allows importation of XMI represented use cases models. The Use Case Writing module also provides a field-based editor for use cases contents specification. Use cases elements are written in a restricted form of natural language. The benefit of a field-oriented form is that use case writers do not have to worry about delimiting the different parts of their use cases. UCEd checks use cases and the domain model against each other and reports inconsistencies and omissions.

The Use Case Composition module implements a finite state transition machine (FSM) generation algorithm described in [2]. The Use Case Composition module analyzes use cases and merges their partial behaviors with a FSM obtained from previously composed use cases. The Use Case Composition module uses information in the domain model to drive the composition process. Some inconsistencies in the use cases and in the domain can be found and reported during use cases composition.

Use cases composition results in a FSM specification that combines all the use cases partial behaviors. Finite state transition machines can be used as prototypes. The Use Case Simulator generates a prototype with a graphical user interface from the FSM specification. Using the interface, UCEd allows "playing" the use cases giving an opportunity to validate requirements and uncover possible interactions between use cases.

References

[1] ArgoUML project. http://argouml.tigris.org.

[2] An approach for the synthesis of state transition graphs from use cases. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP'03)*, volume I, pages 456–462, june 2003.

[3] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison Wesley, 1998.

[4] OMG. OMG Unified Modeling Language Specification version 1.4, 2001.

[5] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1998.