Pattern-Oriented Use Case Modeling

Pankaj Kamthan
Concordia University, Canada

INTRODUCTION

The majority of the present software systems, such as those that run on automatic banking machines (ABMs), on mobile devices, and on the Web, are interactive in nature. Therefore, it is critical to precisely understand, identify, and document the services that an interactive software system will provide from the viewpoint of its potential users. A large and important class of models that these services encapsulate is *use cases* (Jacobson, Christerson, Jonsson, & Övergaard, 1992).

In the last few years, use cases have become indispensable as means for behavioral modeling of interactive software systems. They play a crucial role in various software development activities, including estimating development cost (Anda, 2003), eliciting behavioral requirements, and defining test cases.

It is well known that addressing quality *early* is crucial to avoid the propagation of problems to later artifacts (Moody, 2005). With the increasing deployment of use cases as early artifacts in software process environments, the question of *how* these models should be developed so as to attain high quality arises. In response, this article focuses on the use case modeling process (the *act* of constructing use case models) and, based on the notion of patterns (Appleton, 1997), proposes a systematic approach towards the development of use case models.

The rest of the article is organized as follows. The background and related work necessary for the discussion that follows is outlined. This is followed by the presentation of a pattern-oriented use case modeling process for systematically addressing the semiotic quality of use case models in a feasible manner. Next, challenges and directions for future research are outlined, and finally, concluding remarks are given.

BACKGROUND

In this section, the terminology necessary for the discussion that follows is presented and the significance of quality in the development of use case models is briefly reviewed.

A Primer on Use Case Models

A *use case* models the behavior of a software system, which yields an observable result of value to an actor of the system (Jacobson et al., 1992). In doing so, a use case intends to capture typical interactions between the actors and the software system being built. There can be multiple use cases of a system and they can be related. The use cases can be classified into analysis use cases (those addressing the problem domain) and synthesis use cases (those addressing the solution domain). There are three possible types of (binary and non-reflexive) relationships among use cases: "include," "extend," and "generalization."

An *actor* is an external entity that interacts with each instance of a use case. An actor could be a (human) user or another program. Each actor plays a unique *role* with respect to a use case from the viewpoint of the system. There can be multiple actors of a system, and they can also be related. The actors can be classified into primary actors (those initiating a use case) and secondary actors (those supporting the goal of a primary actor). There is one possible type of (binary and non-reflexive) relationship among actors: "generalization."

A use case is an abstraction of the real-world use of a system. A *scenario* is a concrete realization or instance of a use case; it can be classified as either normal or non-normal (including alternates and exceptions).

The s*ystem boundary* is a means to illustrate the separation of actors and use cases. The set of all actors and use cases describing the complete usage of a system is known as the system's *use case model.*

There are two common means of representing use cases: as structured text and as a graphic. Each means of representation has its own advantages and limitations (Jacobson, 2003); a detailed discussion of this issue is beyond the scope of this article.

The Unified Modeling Language (UML) (Booch, Jacobson, & Rumbaugh, 2005) is a standard language for modeling the structure and behavior of object-oriented software systems. UML provides explicit support for use case models (Bittner & Spence, 2003): the Use Case Diagram is a commonly used UML diagram type to graphically represent use case models, the Activity Diagram is used to represent the sequential order in which use cases are executed, and the Sequence Diagram is used to represent scenarios.

A View of the Quality of a Use Case Model

Using ISO/IEC 9126-1:2001 Standard, the quality of a use case model could be formally but broadly defined as the totality of characteristics of a use case model that bear on its ability to satisfy stated and implied needs.

There are different views of quality (Wong, 2006) of which an intersection of the product, user-based, manufacturing, and value-based views are applicable in our case. That is because a use case model must satisfy certain quality attributes, be communicable to the user, must conform to the language specification it is expressed in, and its development must be feasible.

Related Work on Quality-Centered Use Case Modeling

There are various reasons why quality of a use case model can be compromised, including lack of understanding of the underlying domain, lack of knowledge or skills in the modeling language, or limitations imposed by modeling tools. There have been a few initiatives addressing the quality of textual and graphical use case models (Rosenberg & Scott, 1999; Fantechi, Gnesi, Lami, & Maccari, 2003; El-Attar & Miller, 2006; Törner, Ivarsson, Pettersson, & Öhman, 2006). However, these efforts focus on the product, not on the process.

It was previously suggested that the development of use case models needs to be carried out iteratively (Rosenberg & Scott, 1999; Bittner & Spence, 2003; Leffingwell & Widrig, 2003). However, details are sketchy, and the emphasis on the improvement of quality of use case models is lacking. In particular, there is no use of patterns. A model-driven requirements engineering process that integrates certain metrics for the improvement of quality of use case models has been proposed (Berenbach & Borotto, 2006). However, the rationale for the selection of quality attributes is unclear.

A SYSTEMATIC APPROACH FOR THE DEVELOPMENT OF USE CASE MODELS OF HIGH SEMIOTIC QUALITY

In this section, a use case modeling process for systematically addressing the semiotic quality of use case models is proposed.

A Pattern-Oriented Use Case Modeling Process

The motivation for a use case modeling process (henceforth labeled as UCMP for brevity) is based on the assumption that a "good" process will lead to a "good" outcome of the process (Nelson & Monarchi, 2007), namely the use case model.

Characteristics of a Use Case Modeling Process

UCMP is expected to have the following characteristics:

1. UCMP must be cost-effective. The advantages of adopting UCMP must substantially outweigh the costs.
2. UCMP is a *sub-process* of a user-sensitive software development process. Examples of these include Crystal Methodologies (Cockburn, 2005) and the Unified Process (UP) (Jacobson, Booch, & Rumbaugh, 1999). The processes for domain modeling and requirements elicitation can impact the "velocity" of UCMP.
3. UCMP must be *both* iterative and incremental. They are complementary: being iterative enables re-visitation of a use case model, and being incremental facilitates the progress of a model.
4. UCMP must have *explicit* support for model quality assurance and model evaluation. This is necessary for construction of use case models to attain desirable quality.
5. UCMP should have a quantifiable and feasible stopping criterion.

Use Case Modeling Process, Quality, and Patterns

The reliance on past experience and expertise is critical to any development. A pattern is a proven solution to a recurring problem in a given context (Appleton, 1997). A unique aspect of a pattern is that it not just describes how but *why* a certain solution works, the scope within which it works,

Table 1. A framework for addressing the semiotic quality of use case models using patterns

Product	Semiotic Level	Means for Quality Assurance	Decision Support
Use Case Model	• **Pragmatic:** Maintainability (Modifiability, Portability, Reusability), Usability (Comprehensibility, Readability) • **Semantic:** Completeness, Validity • **Syntactic:** Correctness	Patterns	Feasibility