Chapter VII
Data Modeling and Functional Modeling:
Examining the Preferred Order of Using UML Class Diagrams and Use Cases

Peretz Shoval
Ben-Gurion University of the Negev, Israel

Mark Last
Ben-Gurion University of the Negev, Israel

Avihai Yampolsky
Ben-Gurion University of the Negev, Israel

ABSTRACT

In the analysis phase of the information system development, the user requirements are studied, and analysis models are created. In most UML-based methodologies, the analysis activities include mainly modeling the problem domain using a class diagram, and modeling the user/functional requirements using use cases. Different development methodologies prescribe different orders of carrying out these activities, but there is no commonly agreed order for performing them. In order to find out whether the order of analysis activities makes any difference, and which order leads to better results, a comparative controlled experiment was carried out in a laboratory environment. The subjects were asked to create two analysis models of a given system while working in two opposite orders. The main results of the experiment are that the class diagrams are of better quality when created as the first modeling task, and that analysts prefer starting the analysis by creating class diagrams first.

INTRODUCTION

The main goal of this research is to examine the better order of performing the two main activities in the analysis phase of UML-based software development processes: functional modeling with use cases, and domain (conceptual data) modeling with class diagrams. Though system development is usually an iterative process of refinement, the analysis stage of each iteration should be driven by a specific modeling activity, implying that activity ordering in iterative development is a legitimate and important question. As we show in the next section of this chapter, existing development methodologies differ in a prescribed order of performing these activities: some recommend to start with identifying conceptual classes and continue with developing use cases, using the identified classes or objects, while others suggest to start with developing use cases and continue with building a class diagram based on the concepts appearing in the use cases.

Methodologies starting with creating a domain model by building a class diagram argue that the initial class diagram maps the problem domain and allows describing the functional requirements within a well-defined context. The entities in the class diagram serve as an essential glossary for describing the functional requirements and, since it is an abstraction of the part of the real world relevant for the system, it only rarely changes and can serve as a solid basis for other future systems as well. On the other hand, methodologies starting with creating use cases argue that the classes should be based on the functional requirements, and thus should be elicited from them. One reason for this argument is that creating a domain model before learning the functional requirements can lead to a class diagram that include entities that are out of the system's scope.

We expect that creating a domain model prior to defining the functional requirements with use cases should yield better results, i.e. better class diagrams and use cases. This is because objects are more "tangible" than use cases; analysts can identify and describe more easily the objects they are dealing with and their attributes than the functions or use cases of the developed system. Use cases are not "tangible" and may be vague, since different users may define the expected system functionality in different terms. Repeating Dobing & Parsons (2000), the roles and values of use cases are unclear and debatable. Of course, conceptual data modeling is not trivial either; it is not always clear what is an object, how to classify objects into classes, what are the attributes and the relationships, etc. - but still the task of data modeling is more structured and less complex compared to the task of defining and describing use cases. Besides, the analyst has to create just one class diagram for the system rather than many use cases. While in domain modeling the analyst concentrates only on the data-related aspects, in use-case modeling, the analyst actually deals at the same time with more aspects. Use cases are not merely about functions; they are also about data, user-system interaction and the process logic. Because of the above, it seems to us that starting the analysis process with the more simple and structured task should be more efficient (in terms of time) and effective (in terms of quality of the analysis products). Not only that the first product (the conceptual data model) will be good, it will ease the creation of the following one (the uses cases) since creating use cases based on already defined classes reduces the complexity of the task. In the view of the above, we also expect that analysts would prefer working in that order, i.e. first create a class diagram and then use cases.

The above expectations and assumptions can be supported by both theory and previous experimental work. Shoval & Kabeli (2005) have studied the same issue in the context of the FOOM methodology. According to their experiment, analysts who start the analysis process with data modeling produce better class diagrams than those who start the process with functional modeling. They also found that analysts prefer working in