Advanced Object-Oriented Requirements Specification Methods

Roel Wieringa
Faculty of Mathematics and Computer Science, Free University
De Boelelaan 1081a
1081HV Amsterdam, the Netherlands
roelw@cs.vu.nl

This tutorial presents the latest developments in the field of object-oriented requirements specification and places them into perspective by comparing them to recent developments in structured analysis. The following four techniques and methods are treated:

- the Unified Modeling Language (UML) by Rumbaugh, Booch and Jacobson,
- the 1996 version of Fusion extended with Use cases,
- the 1996 version of OOA (Shlaer-Mellor), and
- the 1993 version of the Yourdon Systems Method.

The treatment is based upon published material as well as information publicly accessible at WWW sites. The presentation thus excludes developments of these methods that are internal to the companies that market the technique or method.

The techniques and methods are analysed in terms of a framework for requirements specifications that is derived from systems engineering. The framework defines several views on software products. First, it distinguishes externally observable software product behavior from internal software product decomposition. External behavior is often described by listing required system functions. Second, the internal decomposition is further divided into a conceptual decomposition, in which the components have a meaning in terms of the environment in which the software product will operate, and a physical decomposition that has a meaning in terms of the implementation on which the software will run. The tutorial is restricted to what the techniques and methods have to say about ways to specify external behavior and the conceptual decomposition, as well as the relationship between the two.

The techniques used by the methods use to specify these different views of a software product are reviewed. Roughly, the 1996 version of the object-oriented methods treated in this tutorial specify external behavior by means of use cases and the conceptual decomposition as a collection of communicating objects. Communication may be synchronous or asynchronous, and each object may perform its behavior according to a life cycle. The 1993 version of the Yourdon Systems Method specifies external behavior by means of a list of events to which the software product must respond, the initiator of the events, the desired system response and the data entering and leaving the product during the event or its response. The conceptual components of the system are data processes, data stores, event stores and control processes. The tutorial goes into some detail to show exactly how external behavior and conceptual decomposition are specified in each of the methods. In particular, the elements in the notations of the methods are listed and compared to each other.

In general, the object-oriented techniques and methods tend to be strong in defining a coherent, modular conceptual architecture for the software product, where the structured methods tend to be strong in the definition of functional requirements on external software behavior. An obvious possibility for combining parts of the two approaches is to use heuristics and techniques from structured analysis for the specification of external behavior requirements and object-oriented techniques for the specification of a conceptual decomposition of the system. It turns out that the structured techniques for specifying external behavior can readily be combined with use case specification, but that the techniques of structured and object-oriented conceptual decomposition are incompatible.

If we compare the techniques for conceptual decomposition of object-oriented methods with those of structured analysis, two major differences stand out. First, structured methods separate data from control, whereas object-oriented methods encapsulate data and control into objects. Second, structured methods encapsulates control around functions whereas object-oriented methods encapsulate control around objects. The first difference may lead to conceptual difficulties when transitioning from structured to object-oriented decomposition. On the other hand, it is shown that the second difference is more apparent than real and should not lead to major conceptual difficulties in the transition.

R.J. Wieringa is associate professor in computer science at the Free University, Amsterdam. He recently wrote a book about Requirements Engineering Methods and Frameworks, published by Wiley, and is currently preparing a second book on Semantic, Real-Time and Object-Oriented Requirements Engineering Methods.

1090-705X/97 $5.00 © 1997 IEEE