Role Modeling Enters the Main Stream

Trygve Reenskaug

Conventional wisdom has been that the Object Modeling Technique (OMT) covers all needs for object-oriented analysis and design. A new wisdom is now emerging: OMT needs to be augmented with an entirely different abstraction to cover the needs of distributed systems and system reuse.

The old conventional wisdom is easily explained. Being implementation oriented, OMT is very helpful for programmers struggling with their source code. The new, emerging conventional wisdom stems from the fundamental weaknesses of OMT. Since it is based on object-oriented programming languages and entity-relationship modeling, it misses that the essence of object orientation is object identity and object collaboration. Therefore, OMT is inadequate for describing systems of interacting objects.

I have recently got involved in the meta-modeling efforts of the Object Management Group (OMG). A profound evolution is clearly in progress. OMG is primarily interested in object interoperability. No wonder they are poorly served by the class oriented approach, and no wonder they are actively searching for something different. At the OMG Madrid meeting in July, I presented our role modeling technology to the Business Object group, and was told that we were very much in line with their own views and requirements.

This year's OOPSLA conference was in San Jose, California in October. Several Birds Of a Feather sessions and chance encounters strengthened the impression that conventional wisdom is on the move. The fundamental mismatch between OMT and Jacobson's Use Cases further supports this view.

The impression was solidified at the November meeting of OMG in Nice, France. The Object Analysis and Design group, the Business Object group, and the Meta-Object group all search for suitable ways of describing systems of distributed objects. An ad hoc Boundary Group shall endeavor to harmonize their different requirements and proposed solutions. All serious proposers, including IBM, Taskon, OPEN, and even Rational, are adding the new dimension to the traditional OMT paradigm.

Conventional OMT supports the *class dimension*. The abstractions are based on the similarity of objects, and the focus is on build-time system aspects. Concepts such as class, type, and various kinds of relations and associations belong to this dimension. For example, class *Bank_Account* could capture what is common to all bank accounts; the derived classes *Deposit_Account* and *Loan_Account* could symbolize possible specializations.

In our *role model dimension*, the abstraction is based on representing a coherent story by a system of objects. We focus on run-time system aspects, describing how objects collaborate to achieve a common goal. A *role* is a partial description of an object, describing its responsibility and other properties relevant to the role model's *Area of Concern*. An *activity* is a composition of object interactions that serve a system goal. A role represents one specific object in an instance of the role model. This makes the model quite concrete and permits us to study its overall behavior. A *Role Collaboration view* shows the object collaboration structure. *Interaction Scenarios*, *Data Flow Diagrams*, and *Finite State Machines* describe the system activities. *Use Cases* are fully supported by identifying and organizing interesting *Areas of Concern*.

For example, role model *Teller_Machine_Withdrawal* could describe how a Customer interacts with his Account through a Teller_Machine to increase his cash on hand and simultaneously reduce the account balance. Role model *Manual_Deposit* could describe how a Customer interacts with his Account through a Manual_Teller in order to achieve the opposite result.

All the static and dynamic properties of a *base role model* can be inherited into a *derived role model*. In this *synthesis* operation, all the roles of the base model are mapped onto corresponding roles in the derived model. We thus inherit not only the properties of individual roles, but also all the base model activities. The result is that object patterns and frameworks become first class citizens in a world of domain standards, software library products, and the reusable assets of enterprises.

These are very powerful reuse capabilities. A generic design pattern can be synthesized into a specific design; while the specific implementation classes are subclassed from a corresponding ensemble of base classes. In 1978, I created the first version of the Model-View-Controller (MVC) framework while being a visiting scientist at Xerox PARC as part of a project on project planning and control. The idea was that the MVC describes the base roles, their responsibilities and joint behavior. The MVC framework can be specialized. For example, a text editor can be constructed by specializing the MVC design and subclassing the MVC classes.

The role model dimension is linked to the class dimension through interface descriptions. A role model activity consists of objects invoking operations in other objects. These operations are described in interfaces associated with the different interaction paths. The interfaces of the corresponding classes are created as the union of all the relevant role model interaction interfaces. Role and class modeling are therefore complimentary, and can be used separately or in combination in a consistent manner.

Until now, OMG has focused on standardizing interfaces. I believe this has been a sound decision, since the only viable alternative has been to specify implementation details such as classes and class relationships.

OMG now needs to take a wider view, and to describe whole systems. They need to describe object structures and their interaction patterns, and they need to do this in the form of generic models that can be specialized according to the concrete needs in different circumstances. They also need to be implementation independent; indeed, they need to be free of the 'single model tyranny' that is underlying all build-time description systems.

(E.g., the 'global schema' of data base systems, and the 'global variables' of programming languages).

Role modeling traces its roots to the early seventies, when we here in Norway started applying object oriented technology to shipyard production planning and control. Since then, we have continuously developed our role modeling methods and tools. We initially did this to satisfy our own software engineering requirements. Other people in telecommunications, engineering, banking, and other industries got to hear about it and persuaded us to package out methods and tools as products for others to use. Our current products are described on our web site http://www.sn.no/taskon/. The tools can be downloaded from this site for evaluation purposes. To us, role modeling is "the common sense of objects".

Many theoreticians and practitioners in OMG and elsewhere see a need for role modeling capabilities. Egil Andersen of the University of Oslo is now in the final stages of establishing a sound theoretical foundation; augmenting its 20 years of practical experience with solid theory. Role modeling is ready for the main stream.

Role modeling and other aspects of object technology is described in my recent book:

Reenskaug, Wold, Lehne: Working With Objects. Manning/Prentice Hall 1996
ISBN 0-13-452930-8