OOPSLA'96 Poster Synopsis:

Dynamic Modeling with DYNAMO.

Kurt Nørmark, Department of Computer Science, Aalborg University, Denmark.

E-mail: normark@cs.auc.dk
WWW: http://www.cs.auc.dk/~normark
Keywords: Object-oriented design, dynamic models, tool support.

Introduction and overview.
Object-oriented design models are usually divided into static and dynamic models. Static models are oriented towards classes and class relationships. In general, static models emphasize the structural properties of a design. Dynamic models deal with objects and object relationships. Dynamic models emphasize the process and temporal properties of a design.

It is difficult to capture dynamic models on a sheet of paper. The main reason is that the set of live object develops as a function of time: objects are created and die while they interact with each other. As a consequence we represent dynamic models in a *dynamic medium*. The medium manifests itself as a set of tools for model construction and exploration. The tool set is called DYNAMO.

In the dynamic models proposed in the OOD literature, much emphasis has been put on the diagrammatic notation, and less on the underlying concepts. In DYNAMO we deliberately de-emphasize the means of expression (text vs. graphs vs. other kinds of diagrams). Instead we develop an abstract representation of a dynamic models. The abstract representation can, of course, be used as the basis for generating the most popular diagrams known from the OOD literature (message trace diagrams or object message diagrams). What is important, however, is that our approach allows us to focus on the conceptual issues of dynamic modeling instead of the more superficial, notational issues.

DYNAMO is scenario-based. In other words, our dynamic models are fixed examples of object interactions, which can be executed in the mind of the designer. Execution in terms of simulation of the model on a computer is not the goal of this work.

There are two central elements in our work with DYNAMO: (1) Definition of high-level design concepts for expressing dynamic models, and (2) development of tools to allow the construction and exploration of the models. We now give a brief overview of both of these. More information (including descriptions of similar work) can be found in existing papers [2, 3] as well as on the WWW home page for the project [1].

The DYNAMO Concepts.
A scenario consists of an *initial scene* of objects and a number of top-level interactions with these. At any point in a scenario the scene is the set of objects that is relevant for the current dynamic model. Objects enter the scene via a mechanism called *object provision*. An object provision is a convenient mechanism that states the relevance, and claims the existence of an object exactly at the time it is needed by the designer. It should be noticed that 'object provision' is not necessarily equivalent with 'object creation'. As a simple and practical convention, all objects on the scene have a unique name, through which they can be referred to during a scenario.

The life-time of an object, and the relationship between the object and other objects, is represented as an *object keeping*. An object keeping is part of an object provision. We operate with objects that are *global*, *local* to an activation of a method, *part of* or *associated from* other objects, and *floating* (akin to dynamically allocated).

A *message*, with possible parameters, is passed from a sender object to a receiver object. The parameters may be existing objects (referred to by their names), object provisions (objects of which we claim the existence at parameter passing time), or informally given values (of basic types that are not related to any class). It is possible to state an assertion that describes the situation just after the parameters have been passed. Passing a message may cause provision of new objects, other local message passings (recursively), and establishing a result (in terms of an existing object, an object provision, an informally given value, or an informally described effect on the program state). As part of the result, it is possible to state another assertion, which describes the situation at the end of the processing of the message.

The DYNAMO Tools.
The DYNAMO tools consist of a set of browsers for construction as well as exploration of dynamic models. Models are constructed with a simple, special-purpose structure editor that builds an abstract syntax tree representation of the model.

During model construction, and relative to a given point p in a scenario, it is important to know the exact set of objects on the scene. The reason is, of course, that the interaction at point p in the scenario takes place among these objects; nothing more and nothing less. This need for knowing the objects on the scene is alleviated by the *scene browser*. The scene browser is based on data flow analysis carried out on the underlying abstract syntax tree.

Given a set of scenarios it is possible to derive substantial aspects of a static model. The reason is the obvious duality between objects and classes, and between messages and methods. It should, however, be noticed that it is difficult to extract good information about specialization/generalization relationships from the dynamic models. Using the DYNAMO tools it is possible to generate class interfaces (along the lines of header files in C++) for all the classes involved in a scenario.

In a more elaborate version of the DYNAMO prototype we would want to support animated versions of message trace diagrams (along the lines proposed in [4]). We would, in addition, like to construct outlines of method bodies, based on all the messages involving a particular method.

References

[1] Kurt Nørmark. The DYNAMO home page. http://www.cs.auc.dk/~normark/dynamo.html.

[2] Kurt Nørmark. Dynamic models in object-oriented design. Technical Report R-96-2005, Department of Mathematics and Computer Science, Institute of Electronic Systems, Aalborg University, February 1996.

[3] Kurt Nørmark. Tools for exploration of dynamic models in object-oriented design. In Lars Bendix, Kurt Nørmark, and Kasper Østerbye, editors, *Proceedings of the Nordic Workshop on Programming Environment Research, NWPER'96*, pages 185–195. Department Computer Science, Institute for Electronic Systems, Aalborg University, R-96-2019, May 1996 1996. http://www.cs.auc.dk/~normark/NWPER96/proceedings/proceedings.html.

[4] Marko Salmela, Marko Heikkinen, Petri Pulli, and Reijo Savola. A visualisation schema for dynamic object-oriented models of real-time software. In Boris Magnusson et al., editor, *Proceedings of the Nordic Workshop on Programming Environment Research, NWPER'94, Lund*, pages 73–86, 1994.