SOOM and Tornado-*
Experience with Database-support for Object-Oriented Applications
Arne J. Berre
Center for Industrial Research
P.O.Box 350, 0314 OSLO 3, NORWAY
uucp: mcvaxlsi.uninett!berre

Abstract
SOOM, Semantic Object-Oriented Model, and Tornado-* are the datamodel and database-architecture supporting the object-oriented Taskon-Environment. The major component in this environment is a set of structure-oriented editors that manipulate a hypertext-like document-structure. SOOM is a datamodel based on a merge of ideas from structural and behavioral object-oriented datamodels, where relations are added to an object-oriented language, Smalltalk-80. Tornado-* is an architecture based on one or more centralized object-servers and a local workspace on each workstation. The current bottleneck of the initial caching from a centralized server to the local workspace is identified, and a solution based on a distributed object-server is suggested. This paper focus on the multilayer approach used in both the model and the architecture.

1. Introduction
Taskon-Environment is an Engineering Information System which has been developed in the research program "Effectiveness and Quality in Engineering" at Center for Industrial Research, in the period 1983 to 1988. The system is based on an object-oriented architecture in a distributed workstation/server environment, [REEN86/]. The main programming language for development of tools and applications has been Smalltalk-80.

The information management in Taskon-Environment is organized around a hypertext-like document-model of interconnected objects. The objects may contain different kinds of information, like text, pictures, drawings, spreadsheets, database-queries etc. The user-interface is managed by a set of structure-oriented editors which uses the structural schema of the document to prevent the user from creating illegal structures. The application-areas for the system have been preparation of bid-proposals in offshore engineering, and information management for software engineering.

The developed datamodel was named SOOM because of its integration of concepts from semantic datamodels and object-oriented programming languages, and the system was named Tornado-* because the Data-Store-part is integrated into our earlier system, Tornado [ULFS82/]. Tornado is a network-oriented database with variable-length records, originally developed for CAD/CAM-applications.

2. SOOM - Semantic Object Oriented Model
The datamodel was developed to support complex document-structures, and is a synthesis of ideas from semantic datamodels and object-oriented languages. An object-oriented model like Smalltalk does not support structural abstractions in the same sense as most semantic datamodels. Typical semantic datamodels support relationships to describe 1-to-1, 1-to-many or many-to-many relations between objects, and this is a useful construct for representing complex object-structures.

As has been pointed out in [BERR86/ and [RUMB87/], the notion of relations is a useful semantic construct which is normally missing in object-oriented languages, but should ideally be supported by an object-oriented datamodel.

The following shows how the structural and behavioral aspects of SOOM are integrated in a multi-layered model.
2.1. Different views on the shared data from different applications

Figure 1 shows different mappings for the same object, from storage-object to structural object to behavioral object, and some temporary display-objects which are used to present different views to the user.

Since different applications might want to present different aspects of a shared structure, there is not a need to share the behavioral part of objects. The structural part represents the information which needs to be persistent. Compared to the ANSI-SPARC 3-schema architecture, the structural schema corresponds to the conceptual schema, the way it is physically stored corresponds to the physical schema, and the aspect of the structural model which is used by an application corresponds to a view on the conceptual model. It is the responsibility of the behavior added in this view to maintain the integrity constraints described by the structural model. This layering gives a freedom to use 1-to-many mappings between conceptual objects and application objects.

The behavioral schema describes the message-interface-protocol for methods applicable to instances of the classes involved in an application. Such an instance is generated based on an instance from the structural model, but can be extended with class-dependent operations, in addition to those inferred from the structural type. The mapping between the structural schema and the behavioral schema is by default one-to-one. An object-type in the structural schema corresponds to a subclass of Model-object in the behavioral schema, and a relation-type corresponds to a subclass of class Relation.

The structural schema can describe 1:1, 1:N and M:N relations between objects. It is important to note that the structural schema describes a structural abstraction which can be realized by different physical data-structures, both in the shared database and in the application-views.