A Method for Constructing Formal Object-Oriented Analysis Model and Supporting Environment

Junichi Furukawa

School of School of Information Science,
Japan Advanced Institute of Science and Technology

February 13,1998

Keywords: Object-Oriented, Formal Model, Supporting Environment.

Recently Object-oriented methodologies are focused and used to develop large-scale systems. OMT is typical of them. OMT consists of three point of views : structural (Object Model), behavioral (Dynamic Model) and functional (Functional Model). Though these three models are desirable for describing systems, syntax and semantics of them are not defined strictly. This makes it difficult to support modeling with computer.

Aoki proposed *Fomal Object − oriented Analysis Model* (called FO∀M) to solve the problem. FO∀M provides independent three formal models (called *basic models*) and a mechanism to unify them (called *interpretation*). These enable us to treat systems formally and make it easy to support with computer. However, only static relations are defined in FO∀M, so a method for constructing models must be defined.

In this paper, a method for constructing models based on conceptions of FO∀M and a supporting environment for FO∀M are proposed. At first, *static relations* among elements in each model are focused. A *static relation* is a relation which must be satisfied finally. For example, when an inheritance is defined in the basic object model, it must be also defined between two identifiers of the classes. Next, *scope rules*, which are need to define interpretation, are focused. Attributions and functions are encapsulated according to their properties in basic object model. Information hiding is realized by *scope rules* which are defined according to structure of these objects. So elements in basic object model must be selected along these rules when interpretation defined. Finally, *order of decision* is defined. There are a lot of ways to construct models. But elements which can be decided after definition of a certain element are limited. Furthermore, there is an order which must be kept whichever way is selected. For example, if an identifier of a certain class is defined, we can decide a relationship between it and identifier of other classes or a set of attributes (or functions) correspond to it.

An environment supporting model construction are designed on these relations and rules. In this environment, *static relations* become indices of completeness of models, *scope rules* become method to give usable elements in basic object model and *order of decision* gives users guidelines on model construction.

It's impossible to grasp and extract whole concepts in the specification when we analyze large-scale systems. So it's desirable to phase and be able to keep information about incomplete models. Hence I introduce relational database system to keep these information. Persistency of data is insured thanks to database system, too. In database, a set of identifiers defined in FO∀M is given as a table. This table must have enough informations to know the element's state and to distinguish each other. Primary key should be declared if a row(s) in the table must have unique value. Now think about a table stands for a set of identifiers of inheritance, for example. An inheritance is a relation between two identifiers of class, so a table of them needs rows for these two identifiers and one more row for it's own identifier. An identifier of inheritance must have unique value, so it will be the primary key of this table.

Because there are possibilities that many items will disappear by deleting only one item, we must be careful when delete items from the database. For example, assume if an identifier of class is deleted, relations refer to that identifier disappear at the same time. This results disappearance of elements, which never need to be deleted, and unexpected collapse of models. So an appropriate policy about deletion must be shaped. To protect this spreading of deletion, I introduce a special identifier $Undef$. If an element which constructs the definition of a relation is deleted, a structure of the relation will still remain. A definition with $Undef$ indicates they are still in incomplete state. So it is possible to promote users to make incomplete parts of models by pointing out the $Undef$.

I show how to construct environment with **Java** as an example. Using **Java** has the advantage of corresponding to multi-platform and good connectivity with database system thanks to **JDBC API**.

We need some guidelines on process of construction along architecture of FO∀M in addition to *static relations* and *scope rule*. The *order of decision* is defined to formalize them. Using relations and rules, defined above, we can get indices and guidelines of making model.

Database system is useful to keep information of models but definition of tables must be done carefully with taking influence of deleting items into account. An influence of deletion is limited by introducing a special identifier, $Undef$. Data stored in database are very important because they will be used in other supporting environment for FO∀M or other phases after analysis.