rules to systematically create a single integrated dynamic logic model from the several separate elements that constitute a description of an object-oriented system expressed in Unified Modelling Language (UML). Rules for mapping the graphic notation onto the formal kernel model are the key component of the transformation method. The information proceeding from each separate UML model element is used to progressively enrich the single integrated MMM specification.

The transformation method will be integrated into a software engineering environment based on compiler architecture, taking advantage of the Rational Case Tool.

3. Concluding remarks

Formal methods are increasingly recognized as a viable technology for the software development process. We recognize the importance attached to the integration of formal methods with existing development practices. We address the problem of gaining acceptance for the use of an unfamiliar formalism by providing sound bases to well known graphical notations.

In this paper we have defined a conceptual object-oriented model that formally represents the information acquired during object-oriented analysis and design.

As other formalization of modeling notations, the principal benefits of the proposed formalization can be summarized as:

- Graphical specifications obtain a precise and unambiguous meaning.
- The different views on a system (class diagrams, sequence diagrams, statecharts, collaboration diagrams, etc.) are integrated in a single formal model. This allows us to define rules of compatibility between the separate views, on syntactical and semantic level.
- Using formal manipulation, it is possible to deduce further knowledge from the specification.
- The faults of specifications expressed using a user-friendly notation can be revealed and removed using analysis and verification techniques based on the formal kernel model.
- The present formalization helps identify ambiguous and inconsistent structures in the UML.
- Formal refinement steps can be defined on model.
- The semi-automatic transformation method will help users and software engineers in the manipulation of the formal notation and also will help them to obtain better-structured and legible formal specifications.

The principal difference between our model and other object-oriented formal models is that the integrated formal kernel language allows the representation of interconnections between model and meta-model elements. This is particularly useful for:

- Description of system evolution; the formal model has to deal with changes at the two levels: Firstly, at the meta-model level, it is necessary to represent changes in the scheme of classes (e.g. adding a class, adding a is-a relationship), changes in class definitions (e.g. adding a method, removing a collaboration). Secondly, at the model level it is necessary to specify object evolution. Lastly, there is the consequent problem of specifying consistency between a set of objects and a set of class definitions that can change.
- Formal description of contracts [Helm and Holland 90] and reuse contracts and reuse operators [Steyaert et al. 96]. A Reuse contract documents how a set of classes can be reused and a reuse operator documents how classes are actually reused (for instance, by refinement, extension, concretization, etc.).
- Description of design patterns [Gamma et al. 96].
- Quality assessment mechanisms: there must be a formal definition of desirable properties for the information captured in the analysis and design activities. The desirable properties must represent those characteristics that ultimately lead to more efficient, usable, maintainable, and extensible software products.
References


