

# Difficulties in Defining Precise Semantics for UML

Ileana Ober

Telelogic Verilog  
150, r. Vauquelin – BP 1310 31081 Toulouse - France  
Email: [ileana.ober@csverilog.com](mailto:ileana.ober@csverilog.com)

Institut National Polytechnique de Toulouse  
Laboratoire d'Informatique et Mathématiques Appliquées  
2, rue Camichel – BP 7122  
31071 Toulouse - France

## 1. How to add precise semantics to UML

The attribute “*precise*” is appealing and almost anyone will admit needing a *precise* UML. However, depending upon background and needs, people perceive *precise* differently. A mathematician will probably understand something very formal without a single possibility of misunderstanding or interpretation, whereas an analyst will probably ask for less.

### 1.1. Semantics with flavors

UML is used in a wide area of contexts, by people coming from different cultures, many of them considering (more or less justified) their case special and asking for a deviation from the standard in the form of a particular tuning of UML. A “hard-coded” UML precise semantics would preclude the existence of these tunings and thus would be practically unacceptable.

We are in a situation where on one hand we need precise semantics to really make the UML a communication mean, while on the other hand we do not need too much precision because the domains on which UML is to be applied are so different that they can not be unified under a single semantics. One of the proposed outcomes of this situation was made in a previous pUML workshop [FAQ] and consists in providing the semantics as part of *prefaces* UML model complements, which give it a precise meaning.

This is more or less, the approach currently taken by the OMG [RQP], with the introduction of *profiles* as standard means to adapt the UML to some domain-specific needs.

Both the *profiles* and the *prefaces* still raise two major questions with respect to semantics: the granularity problem (what do we need to have in the basic UML, and what in the profile) and the profile refinement problem (can we have hierarchies of

profiles, and if so, how are they defined and what constraints should be imposed on them).

## **1.2. The granularity problem**

As we have seen, part of the precise semantics should be contained in domain-specific parts, i.e. profiles or prefaces. The question that naturally arises from this is what do we put in the basic UML and what do we put in the domain specific parts?

### **1.2.1. Semantics defined in profiles**

A possible approach is to leave the UML definition as it is today. Each profile should contain all the semantics that describes it. The advantage is that this allows almost any “UML semantics” to exist.

This approach does nothing for increasing the UML precision, it only gives semantics to UML dialects. One of the main goals of asking for a precise UML, which is to ensure that UML offers a communication means between modelers, is compromised, as the same UML model may be understood differently in different contexts.

Actually, this approach would lead to the transformation of UML from a modeling *language* to a modeling *paradigm*. If no concept had any semantics, then UML would only be a vocabulary of terms with different meanings in different contexts.

Concretely this approach would consist of leaving the UML definition as it is today (possible removing inconsistencies and omissions, if they are found), and adding precise semantics into UML profiles.

### **1.2.2. Semantics shared between profile and meta-model**

Another approach would be to add semantics into profiles, but to also add more information into the UML definition.

The concrete way to do it, is not obvious and a discussion during the workshop could be benefic.

A possible solution would be to define the concepts, relationships between them, constraints, some more precise semantics of them in the *common* UML. Moreover, for each concept it is stated explicitly in the *common* UML whether it can or not be refined (or redefined) in a profile. The advantage is that the impact of different variants is reduced and localized and anytime someone will look at a UML model it will be clear which elements are susceptible of having a semantics different from the *common* one.

In the same spirit we could imagine not only having concepts whose semantics can be refined, but also to having without semantics, so that any UML profile should clearly state what is the meaning of that concept in its context.

Other intermediate alternatives could be imagined and this may represent a discussion topic during the workshop.

### 1.3. Flavors of the flavors

Assuming we know how to partition the semantic between the *common* UML and the specific *profile*, the next questions that comes is how much information the couple common UML plus specific profile should contain?

We take as a concrete example the under-work UML profile for real-time [RFP-RT], as it is one of the first profiles demanded by the OMG. The purpose of the RT profile is to offer specific means appropriate for the real-time domain. Although it solves many of the UML problems, it still may need further refinement. The real-time domain is itself vast, and a single profile could not address every specific demand. As a result, if one would compare the UML profile for real-time with a solution dedicated to a specific real-time field, such as the SDL [SDL] primary designed for telecommunication, the conclusion would be that the real-time profile offers less than the existing solutions and may still need further refinements.

The conclusion that can be drawn is that we would need mechanisms for hierarchically defining the profiles to further adapt them for sub-domains.

## 2. How precise the precise semantics should be?

As stated in [EVA99] there are several reasons one may need precise semantics. Some of them are:

- to ensure efficient communication means;
- to provide a notion of well-formed model;
- to enable the existence of advanced features in tools (symbolic simulators, test tools, etc).

According to the actual reason for which one needs the semantics, the level of precision of the needed semantics is different.

Although not a goal as such, a formal semantics would facilitate the development of some advanced features in tools, and powerful analysis of models. However, a formally described semantics has the disadvantage that it is more difficult to be read than an informal one. Note however that, even if a formal semantics would be defined, it would not replace the current informal definitions, and the “normal” modeler will not have to read it, all the benefits of it being hidden by tool features.

An important issue related to the semantics definition is the semantic correctness. The semantic correctness could raise discussion for its own, still for now on we will consider that semantic *correctness* is achieved when we have a semantic which is both *complete* and *not contradictory*. Formal semantics is not a guarantee of correctness. Nevertheless using a formalism for the semantics definition, it might be easier to check correctness as there may be tools that allow to check the correctness of the semantics definition.

### **3. What should the dynamic semantics describe?**

Currently the UML offers several means of describing behavior:

- 1.

## 4. An effort to define UML executable semantics in ASM

Tools are the way most people interact with UML. Therefore one important concern is to help tools offer as much support as possible to the UML user. With this usability concern in mind, we have started the definition of an executable semantics for UML, which would provide both a formal precise semantics for the static concepts of UML and a precise definition of the run-time behavior.

We define the semantics on abstract state machines (a.k.a. evolving algebra), which are a readable, easy to use and expressive formalism. They have been used for the definition of the new formal semantics of SDL, and therefore they present the advantage of already having been applied successfully on a similar problem. In addition to this, there is readily available tool support for ASM that allow to check and generate code from ASM specifications.

We have started the definition of an executable semantics with identifying the subset of UML on which to start defining the semantics. The ASM specification of semantics is easily extensible therefore starting the definition of semantics with a subset of UML does not preclude further extensions of the UML areas formalized.

The attempts to formalize UML should take into account the information contained in the UML meta-model, and in the *Well Formedness Rules* contained in the Semantics document. In order to do this, we have developed an ASM code generator from meta-model specification. We have also translated the existing WFRs into ASM constraints.

As the final goal of this approach is to get to an executable semantics of UML, we have also initiate the work on the definition of the dynamic semantics. This work is still ongoing, but it allows us to perceive a future easy to read, flexible executable semantics, which could represent the basis of future tool development.

## 5. Conclusions

Although the problem of not having a precise semantics has been often signaled, having a precise semantics of UML is still an aspiration. This is due partially to the fact that there are still no settled answers to some preliminary questions on the need of precise semantics: how precise do we want the semantics to be, how to much semantics to add in the standard UML and how much to the profiles, do we also need a formal semantics or not, do we need an execution semantics, etc.

There are already some results [EVA99][BOE00], that let us think that we may have a precise semantics that would enable advanced features in tools, and tools interoperability.

We have tried to give some answers to the still open questions in a pragmatic way, by concretely proceeding an executable semantics definition. The work is still ongoing, but the preliminary results are encouraging as the obtained formal description is quite simple.

As the work is ongoing, it would benefit of some possible discussions on the still open questions regarding the UML semantics itself and the way to add more semantics to UML.

## References

- [BOE00] E. Borger, A. Cavarra, and E. Riccobene, *An ASM semantics for UML activity diagrams*, AMAST 2000
- [EVA 99] A. Evans, S. Kent. *Meta-modeling semantics of UML: the pUML approach*. 2nd International Conference on the Unified Modeling Language, LNCS 1723, 1999.
- [FAQ] Stuart Kent, Andy Evans, Bernhard Rumpe : *UML Semantics FAQ*, Results of the pUML workshop at ECOOP 1999, Lisbon
- [RFP-AS] Object Management Group (OMG), *Action semantics for UML Request for Proposal*, OMG Document ad/98-11-01 [ftp.omg.org/pub/docs/ad/98-11-01](ftp://ftp.omg.org/pub/docs/ad/98-11-01)
- [RFP-R] Working response to the OMG RFP For Scheduling, Performance, and Time
- [RT-RFP] *UML™ Profile for Scheduling, Performance, and Time* - Request for Proposal  
OMG Document: ad/99-03-13
- [RQP] Object Management Group (OMG), *Requirements for UML Profiles*, OMG Document ad/99-12-32
- [SDL] ITU-T, Recommendation Z.100, *Specification and Description Language (SDL)*, <http://www.itu.ch> - Electronic Bookshop, Geneva, 1999
- [UML99] Object Management Group (OMG), *UML 1.3 Documentation Set*, [ftp.omg.org/pub/docs/ad](ftp://ftp.omg.org/pub/docs/ad), 1999.
- [WEB-AS] <http://www.umlactionsemantics.org>