Slicing the Three-layer Architecture: A Semantic Foundation for Behavioural Specification

Michelle L. Crane

Queen's University, Kingston, Ontario, Canada

Abstract. We outline a research proposal whose overall goal is to contribute to the definition of a formal semantics for UML, and indeed visual behavioural modelling languages in general. Specifically, we aim to validate the three-layer semantic architecture, used as a way of explaining the behavioural semantics of UML. The validation includes a definition of the semantics of UML actions and activities, as well as a prototype interpreter.

1 Motivation

Languages are used for the communication of ideas. This communication becomes compromised when the various participants in the communication do not fully understand, or agree upon, the semantics of the language. Visual modelling languages, especially those using common elements, such as text boxes and connecting arrows, can be very susceptible to this kind of miscommunication. Consider, for instance, that a state machine diagram could have three different interpretations, depending on the formalism used to interpret it [5].

It is important to formally, i.e., precisely and unambiguously, define the semantics of visual modelling languages. Developers must understand the semantics of the language being used in order to communicate with users and other developers. Tool vendors must understand the semantics of the language so that their tools support the same semantics understood by developers. In addition to automatic manipulation, transformation and analysis of models, tool interoperability is becoming more important in Model Driven Development (MDD). Tool chains, composed of tools created by different vendors, are linked together to provide customized solutions for developers. These tools must be able to communicate with each other, i.e., they must use a common semantics.

Although visual modelling languages are becoming increasingly widespread, formally defining their semantics has proven difficult. There are several reasons why it is harder to define the semantics of visual languages than textual ones:

- Visual languages are two-dimensional, unlike a sequence of characters. There are many factors of a diagram that might or might not be relevant to its meaning, such as proximity of elements to each other, size/shape/colour of elements/lines/arrowheads, etc.

– There is a tendency for people, including developers, to view diagrams as merely "doodles" [6], increasing the likelihood that syntactic and semantic constraints are violated.
– Many visual languages make use of common elements, e.g., text boxes, arrows, etc. Because these constructs are so common, there may be a tendency for readers to automatically assume that they understand the language.

A language is defined by its notation (syntax), together with the meaning of syntactic elements (semantics). This meaning is expressed by mapping the syntax to a semantic domain [6], usually one that is well-understood. The Unified Modeling Language (UML) is a "general-purpose visual modeling language" [11] that can be used in the analysis and design of software systems. Although well-documented (over 1000 pages of specification), UML does not yet have a formal semantics. The abstract syntax is carefully laid out in the specification, but the meaning of the syntactic elements is discussed in prose, with a smattering of Object Constraint Language (OCL) constraints. The run-time semantics of UML is defined as a "mapping of modelling concepts into corresponding execution" [10]. This semantics is not formally defined in the UML specification [10]. Instead, the specification outlines a three-layer semantics architecture, which "identifies key semantic areas...and how they relate to each other" [10].

This three-layer architecture is shown in Fig. 1. Each layer depends on those below it, but not vice versa. The bottom layer represents the structural foundations of UML, including concepts such as values, objects, links, messages, etc. The middle layer is the behavioural base, which contains mechanisms for individual object behaviour, as well as behaviour between objects. More importantly, this base also contains the description of a set of UML actions. The top layer represents different behavioural specifications in UML, all of which rely on the behavioural base. For instance, activities, state machines, and interactions all make use of the actions in order to express behaviour. These actions are explained in terms of constructs in the structural foundations.

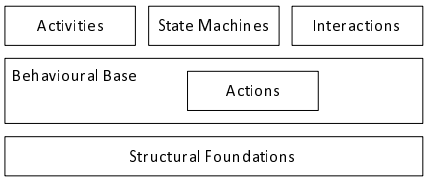

Fig. 1. The UML semantics layers: the Semantics Foundation consists of the bottom two layers

The key to this architecture lies in the middle layer, i.e., actions. A fundamental premise of UML behavioural semantics is the assumption that "all behaviour...is ultimately caused by actions" [10]. Actions are "fundamental units of

behaviour" [10]. As an analogy, actions are comparable to "executable instructions in traditional programming languages" [10].

The advantage of this approach to defining the run-time semantics is the fact that once UML actions are clearly mapped to the structural foundation, it should be relatively easy to define the semantics of different behavioural formalisms.

The three-layer architecture, recently added to UML, has been neither elaborated nor validated. It appears to provide an appealing skeletal framework; however, essential pieces are still missing before it can be leveraged for the definition of a formal semantics for behavioral specification in UML. There is no formal specification of the structural foundations. There is no clear mapping of UML actions to that foundation. Finally, there is no clear mapping from behavioural formalisms to individual actions.

2 Research Hypothesis and Proposed Solution

Hypothesis: The layered semantic architecture is a valid approach for formally defining and implementing behavioural semantics in UML.

We propose to demonstrate the validity of this hypothesis by accomplishing the following tasks:

1. Model the System Model (see Section 3) as a set of UML classes and implement in Java. These classes can be instantiated to create a specific System Model instance, which is a large state machine. Progress: 65% complete.
2. Define a formal semantics for the behaviour of individual UML actions. The semantics will be axiomatic; pre- and post-conditions will be defined on the structural foundations described by the System Model. Progress: 15% complete.
3. Model individual actions as UML classes, and implement in Java; the actions will cause changes of state in the System Model's state machine. Progress: 5% complete.
4. Define a formal semantics for the behaviour of a subset of UML activities. The semantics will be operational; the execution of activities will be described in terms of the execution of individual actions. Progress: 5% complete.
5. Model the subset of activities as UML classes, and implement in Java; activities will be used to combine individual activities into behaviours, which will cause sequences of changes to the System Model's state machine. Progress: 0% complete.
6. Implement a basic prototype activity/action "virtual machine/interpreter", based on the formal semantics. This phase will include the description of a simple surface language for activities. Progress: 15% complete.
7. Look for problems with, or potential improvements to: the three-layer architecture, the current specification of actions and activities, and the System Model. Progress: 5% complete.

3 Related Work

The System Model [2–4] was created as the "basis for a semantic model for UML 2.0"; its goal is to represent the structural foundations of UML (lowest layer of the three-layer semantic architecture from Fig. 1). The System Model is described in a precise fashion, using mathematics, specifically, sets, relations, and functions. By choosing pure mathematics to describe UML, the authors have aimed to avoid limitations or biases inherent in other formal specification formalisms.

One of the basic premises of UML semantics is that all behaviour is "ultimately caused by actions" [10]. Actions were originally introduced into UML 1.5, and in UML 2.0 activities were brought more in line with these actions to produce a more procedural model [11]. UML 2 includes 45 primitive actions to model the manipulation of objects and links, model communication, and model computation [11]. Because UML actions are relatively new, there is little current research on formally defining their semantics. One of the best sources of information about UML actions is a series of articles by Conrad Bock (see [1]), which explain (in natural language) how actions should execute. Research is currently being performed on the concept of executable UML (xUML) [9], focusing on determining a subset of UML, including actions, that could be used to create executable models.

In UML, an activity "represents the execution of a computation" [11]. Defining the semantics of UML activities is a vibrant research area; although the fact that activities substantially changed with the introduction of UML 2 narrows the field somewhat. However, there has been research on defining the semantics of UML 2 activities based on several formalisms, such as Abstract State Machines [12], Petri nets [13], dynamic meta modelling [7], etc.

Our work can also be compared to other research that executes activity diagrams. ActiveChartsIDE [12] is a particularly well-developed project (using Microsoft Visio, Visual Basic, and C#), with an interpreter that allows the simulation and debugging of activity diagrams. Another project is a plug-in for IBM's Rational Software Architect, which permits the animation and debugging of activity diagrams [8]. Neither of these two approaches make use of the UML actions as fundamental units of behaviour. As of yet, there are no published accounts on the formal semantics of individual UML actions.

4 Contributions

The proposed research advances the field of formal semantics of UML in the following ways:

1. Modelling and implementing the System Model will allow us to find inconsistencies, ambiguities and gaps in both the System Model and UML specification, as well as determine if the System Model is over-specified.
2. We will define a formal semantics for (a majority of) UML actions.

3. We will define a formal semantics for basic UML activities, including those containing sequential composition, fork/join, and decision/merge nodes.
4. An interpreter of a subset of actions and activities will be created. This interpreter will implement the formal semantics of actions and activities.
5. We will examine the thesis that the three-layer architecture is a suitable framework for defining the behavioural semantics of UML. By creating our prototype implementation, we will demonstrate that the architecture is valid.

5 Conclusion

The overall goal of this research is to contribute to the definition of a formal semantics for UML, specifically UML actions and activities. When complete, our results will be forwarded to the authors of the System Model, as well as the authors of the actions and activities chapters of the UML specification, in order to improve those documents. In addition, we will also submit our results to conferences related to modelling and formal semantics.

References

1. C. Bock. UML 2 activity and action models, part 6: Structured activities. *Journal of Object Technology*, 4(4):43–66, 2005.
2. M. Broy, M.V. Cengarle, and B. Rumpe. Semantics of UML – Towards a System Model for UML: The Structural Data Model. Tech. Report TUM-I0612, TUM, 2006.
3. M. Broy, M.V. Cengarle, and B. Rumpe. Semantics of UML – Towards a System Model for UML: The Control Model. Tech. Report TUM-I0710, TUM, 2007.
4. M. Broy, M.V. Cengarle, and B. Rumpe. Semantics of UML – Towards a System Model for UML: The State Machine Model. Tech. Report TUM-I0711, TUM, 2007.
5. M.L. Crane and J. Dingel. UML vs. classical vs. Rhapsody statecharts: not all models are created equal. *Software and Systems Modelling*, 2007. Online first DOI: 10.1007/s10270-006-0042-8.
6. D. Harel and B. Rumpe. Meaningful modeling: What's the semantics of "semantics"? *IEEE Computer Magazine*, 37(10):64–72, 2004.
7. J.H. Hausmann. *Dynamic Meta Modeling*. PhD thesis, University of Paderborn, Oct 2005.
8. A. Kirshin and D. Dotan. UML model simulator. Poster handout at MoDELS 2006, Oct 2006.
9. OMG. Semantics of a foundational subset for executable UML models. Initial Submission ad/06-05-02, 2006.
10. OMG. Unified Modeling Language: Superstructure version 2.1. Document ptc/06-01-02, Object Management Group, Jan 2006.
11. J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, second edition, 2005.
12. S. Sarstedt, J. Kohlmeyer, A. Raschke, M. Schneiderhan, and D. Gessenharter. ActiveChartsIDE. Poster at the European Conference on Model Driven Architecture (ECMDA 2005), Nov 2005.
13. H. Störrle and J. Hausmann. Towards a formal semantics of UML 2.0 activities. In *Proceedings German Software Engineering Conference*, volume P-64 of *LNI*, pages 117–128, 2005.