Towards a Rigorous Semantics of UML Supporting its Multiview Approach[⋆]

Extended Abstract[⋆⋆]

Gianna Reggio, Maura Cerioli and Egidio Astesiano

DISI–Dipartimento di Informatica e Scienze dell'Informazione,
Università di Genova, Via Dodecaneso, 35, 16146 Genova, Italy,
`e-mail: {reggio,cerioli,astes}@disi.unige.it`

The task of providing a rigorous semantics of the UML, [9], is far from trivial. Indeed, the UML notation is complex, including a lot of heterogeneous notations for different aspects of a system, possibly described at different points in the development process. Moreover, its informal description is incomplete and ambiguous, not only because it uses the natural language, but also because the UML has the so called *semantics variation points*, that are constructs having a *list* of possible semantics, instead of just one.

A UML model consists of a bunch of diagrams of different kinds, expressing properties on different aspects of a system. Thus a UML model plays the role of an axiomatic specification, but in a more pragmatic context.

Another analogy that we can establish between UML models and specifications is the fact that the meaning of each diagram (kind) can be given in isolation, as well as the semantics of each axiom, and its effect on the description of the overall system is to rule out some elements from the universe of all possible systems (semantic models). Indeed, both in the case of a UML model and of a collection of axioms, each individual part (one diagram or one axiom) describes a point of view of the overall system.

Therefore, our understanding of the optimal form of a semantics for the UML is illustrated in Fig. 1.

We have a box representing a UML model, collecting some diagrams of different kinds, and its overall semantics, represented by the arrow labelled by $\mathcal{SEM}$-UML, is a class of UML formal systems. But, each diagram in the model has its own semantics (denoted by the indexed $\mathcal{SEM}$), that is a class of appropriate structures, as well, and these structures are imposing constraints on the overall UML formal systems, represented by lines labelled by the indexed $\models$. A sort of commutativity on the diagram has to hold, that is the overall semantics must be the class of UML formal systems satisfying all the constraints imposed by the individual semantics. Moreover, the formal semantics must be a rigor-

[⋆] This research has been partially supported by ESPRIT working group 29432 (CoFI WG) and by Programma di Ricerca Scientifica di Rilevante Interesse Nazionale Saladin (Software Architectures and Languages to Coordinate Distributed Mobile Components).

[⋆⋆] This is the extended abstract of a paper presented at AMILP 2000, see [5].

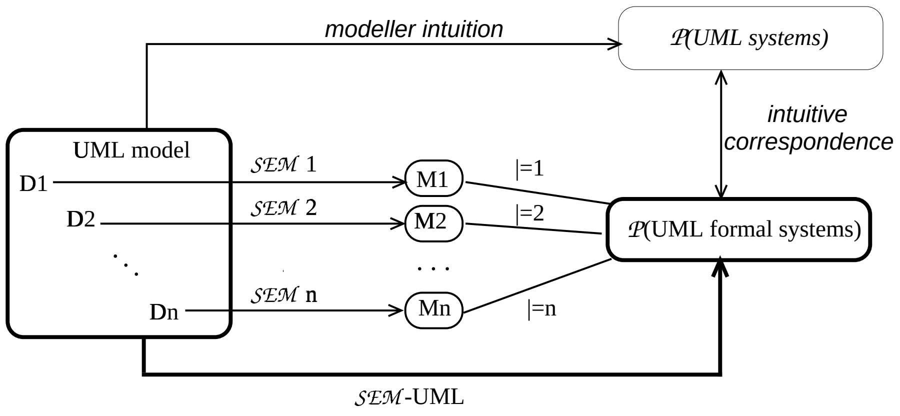

Fig. 1.

ous representation of the expected "intuitive semantics", described by the UML standard, version 1.3 ([9], shortly written from now on UML 1.3).

Several attempts at formalizing the UML are currently under development, but most of them are taking into account only a part of the UML, with no provision for an integration of the individual diagram semantics toward a formal semantics of the overall UML models; see the book [1], and the report [7] on a recent workshop on the topic of the UML semantics also for more references. The only exception known to us is the attempt at describing the semantics of the UML within the UML itself (the *meta-model* approach) as advocated by the pUML group, see the site `http://www.cs.york.ac.uk/puml/`. But even in this case it is difficult to recognize the nature of the semantics of the individual diagrams, as the semantics is given as a sequence of translations into more and more restricted core languages, that are subsets of the UML, and only the smallest is directly given a semantics.

Our approach, accordingly with the previous discussion, is an attempt at formalizing UML models as a whole, while simultaneously giving also a formalization of each kind of diagram in an integrated way. Because of the need of integrating the formalization of both the static and the dynamic part of UML, we have found convenient to use an extension of the CASL basic language [8], namely CASL-LTL [3], especially devised to model formally also the dynamic behaviour of objects. In Fig. 2, we graphically summarize our proposal.

From a technical viewpoint, we proceed in two steps: first, we determine the needed semantic structures (the M_i and the UML formal systems in Fig. 2) through an analysis of the document UML 1.3, and formally describe them. Then, we translate the diagrams into CASL-LTL specifications (represented by the downward arrows), whose formal semantics gives, by composition, the semantics of each diagram in the UML model (represented by the dotted horizontal arrows).

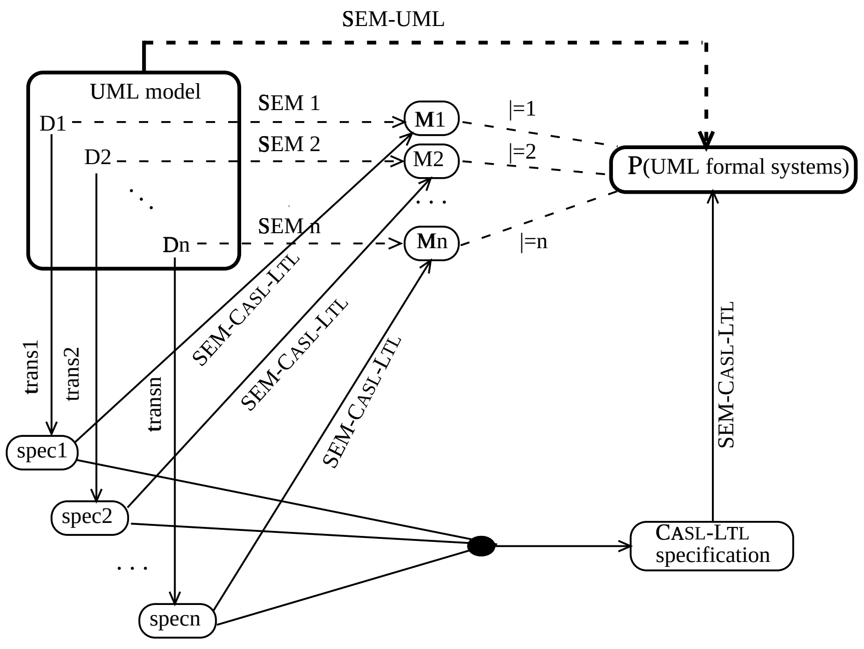

Fig. 2.

Moreover, in the lower part of the diagram, the CASL-LTL specifications representing the individual diagrams are combined (in a non-trivial way) into an overall specification, whose semantics is (has to be) compatible with the constraints imposed by the individual diagrams and provides a semantics for the overall UML model. This combination is graphically represented by a bullet.

We are currently working on filling the above schema, providing the semantic structures and the translations of the various diagrams into CASL-LTL. This activity is performed as part of the `CoFI` [1] initiative, within the `CoFI`-reactive task group.

Roughly speaking, there are two aspects of a system that we are able to describe using the UML: the *structure* of the system, that is which are the components and which are their capabilities, and the *activity* of the system, that is the evolution of its components along the time and the interactions of the system with the external world (e.g., with the users). Since the handling of the time in UML, also of the real time, does not require to consider systems evolving in a continuous way, we have to describe a *discrete* sequence of moves, each one of them representing one step of the system evolution.

Therefore, we will use *generalized structured labelled transition systems* (shortly *glts*) as UML formal systems, representing the evolution steps as transitions.

[1] See the site `http://www.brics.dk/Projects/CoFI`.

Moreover, the labels of the transitions capture interactions with the external world, and the structured states, as sources and targets of the transitions, provide a formal counterpart to the system structure. The components of the structured states corresponding to instances of active classes are in turn modelled by a (non-structured) labelled transition system. Finally, the extra information, like the current stimuli set, are represented, generalizing the standard notion of labelled transition system.

For instance, let us consider a fragment of an invoice system. We have some passive classes, recording information about clients, products (we do not detail these parts), current (and past) orders and stock of an e-commerce firm, and some active classes, managing the above described data and representing two kinds of "software" clerks: the stock handler, who puts the newly arrived products in the stock and removes the correct amount of products to settle an order, and the invoicer, who processes orders and sends invoices. Thus, in Fig. 3 we graphically represent a transition of the corresponding UML system.

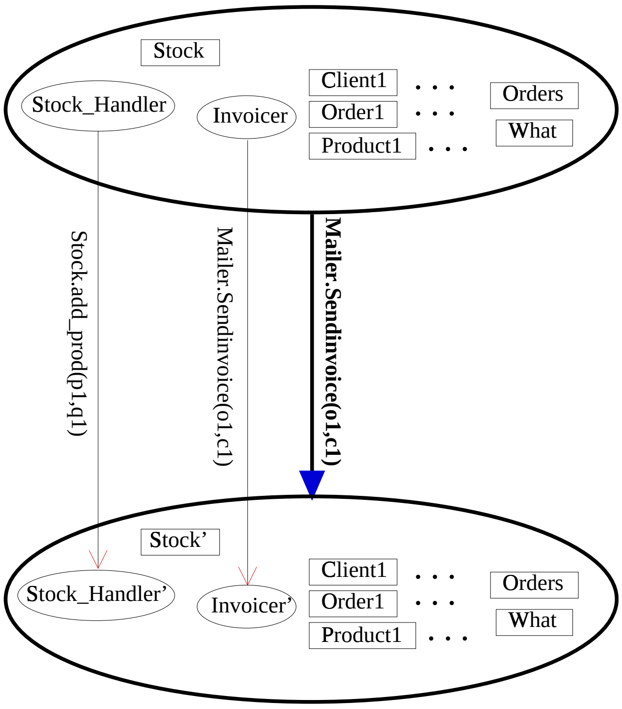

Fig. 3.

Both in the source and in the target state we have two active components (represented by oval shapes) that are, respectively, the `stock_handler` and the `invoicer` and a number of passive components (represented by rectangles): some instances of `clients` and `products`, the unique instance of `stock` and the state of the associations `orders` and `what`, relating each order respectively to the ordering client and to the ordered product. In this picture we consider the parallel execution of two activities, graphically represented by the thin lines connecting the involved active components of the source and target state:

- the `stock_handler` adds to the `stock` some quantity `q1` of the product `p1`; the effect of this action is to change the state both of the `stock_handler` and of the `stock`;
- the `invoicer` send to the client `c1` an invoice for the order `o1`, by calling a method of some external `mailer`; thus, the effect of this action is to change the state of the `invoicer` and to communicate with the external world through the label of the action;

The parallel composition of these actions is described by a transition, graphically represented by the thick arrow connecting source and target structured states. Notice that, since the labels of the structured system carry only information about the interactions with the external world, the label of this transition is taking into account only the call to the mailer. But the resulting state of the system is modified because the internal states of all the objects involved in the move are (possibly) modified. The kinds of diagram considered so far are

- the class diagrams, analyzed and translated into CASL (that is a subset of CASL-LTL) in [2]. Each class diagram describes a data structure, having sorts for each class, functions for getting the attribute values in a given state, and predicates to represent both operations (in order to provide some level of non-determinism needed for parallel executions) and associations. Such information imposes requirements on a UML formal system requiring, for instance, that the states of the system components are elements of the sorts representing classes in the data structure, and that the sequences of transitions corresponding to the execution of an operation connect states that are also connected by the predicate representing that operation.
- the statechart diagrams, analyzed and translated into CASL-LTL in [4]. Each statechart associated with an active class is modelled by a (non-structured) labelled transition system. A UML formal system satisfies such a labelled transition system *lts* if and only if the labelled transition system of the component associated with the active class in the UML formal system is *lts* itself.
- the sequence diagrams, currently under development in [6]. Each sequence diagram is modelled by an *event structure*, where events are sets of UML stimuli, that is, a partial order on sets of UML stimuli. A UML formal system satisfies such an event structure if there is a chain in the partial order that is a path in the transition tree.

Some other kinds of diagrams that we have partly analyzed, and, we conjecture, can be added to our schema without major problems, are

- the collaboration diagrams, as they are rather similar to the sequence diagram;

– the activity diagrams, as they are a specialization of the statechart diagrams.

Moreover, we still have to take into account the deployment diagrams, though we do not foresee particular problems for their formalization within our framework, while we are doubtful about the possibility of giving a formal semantics to the use case diagrams, because they are, roughly speaking, too close to natural language descriptions.

We translate diagram annotations as well, currently using the OCL constraints, but we are in some sense parametric w.r.t. such annotations, so that we could easily substitute any other constraint language for OCL.

The mechanisms for self-extension provided by the UML, like stereotypes, are still to be taken into account.

An extended version of the present paper has been published in the AMILP 2000 proceedings ([5]).

References

1. R. France and B. Rumpe, editors. *¡¡UML¿¿'99 - The Unified Modelling Language.* Number 1723 in Lecture Notes in Conputer Science. Springer Verlag, 1999.
2. H. Hussmann, M. Cerioli, and H. Baumeister. From UML to CASL (Static Part). Technical Report DISI-TR-00-08, DISI – Università di Genova, Italy, 2000. `CASLUMLStatic1.ps` in `ftp://ftp.disi.unige.it/person/CerioliM/`.
3. G. Reggio, E. Astesiano, and C. Choppy. CASL-LTL : A CASL Extension for Dynamic Reactive Systems – Summary. Technical Report DISI-TR-99-34, DISI – Università di Genova, Italy, 1999. `ReggioEtAl199a.ps` in `ftp://ftp.disi.unige.it/person/ReggioG/`.
4. G. Reggio, E. Astesiano, C. Choppy, and H. Hussmann. Analysing UML Active Classes and Associated State Machines – A Lightweight Formal Approach. In *Proc. FASE 2000 - Fundamental Approaches to Software Engineering*, number 1783 in Lecture Notes in Computer Science. Springer Verlag, Berlin, 2000.
5. G. Reggio, M. Cerioli, and E. Astesiano. An Algebraic Semantics of UML Supporting its Multiview Approach. In *Proc. of 2nd AMAST workshop Algebraic Methods in Language Processing (AMILP 2000)*, number 16 in Twente Workshop on Language Processing, Enschede, The Netherlands, 2000. Available at `ftp://ftp.disi.unige.it/person/ReggioG/ReggioEtAl100a.ps`.
6. G. Reggio and A. Mutti. Analysing UML Sequence Diagrams – A Lightweight Formal Approach. Technical report, DISI – Università di Genova, Italy, 2000. In preparation.
7. S.Kent, A.Evans, and B. Rumpe. UML Semantics FAQ . In A. Moreira and S. Demeyer, editors, *ECOOP'99 Workshop Reader*. Springer Verlag, Berlin, 1999.
8. The CoFI Task Group on Language Design. Formal Methods '99 - CASL, The Common Algebraic Specification Language - Summary. Available on compact disc published by Springer-Verlag, 1999.
9. UML Revision Task Force. *OMG UML Specification*, 1999. Available at `http://uml.shl.com`.