From Conditional Specifications to Interaction Charts

A Journey from Formal to Visual Means to Model Behaviour

Egidio Astesiano and Gianna Reggio

DISI – Università di Genova, Italy
{reggio,astes}@disi.unige.it

Abstract. In this paper, addressing the classical problem of modelling the behaviour of a system, we present a paradigmatic journey from purely formal and textual techniques to derived visual notations, with a further attention first to code generation and finally to the incorporation into a standard notation such as the UML.
We show how starting from CASL positive conditional specifications with initial semantics of labelled transition systems, we can devise a new visual paradigm, the interaction charts, which are diagrams able to express both reactive and proactive/autonomous behaviour.
Then, we introduce the executable interaction charts, which are interaction charts with a special semantics, by which we try to ease the passage to code generation.
Finally, we present the interaction machines, which are essentially executable interaction charts in a notation that can be easily incorporated, as an extension, into the UML.

Keywords: design of visual notations, formal notations, behaviour modelling/specification, CASL, UML, interaction charts

1 Introduction

In a remarkable paper [10], celebrating and assessing a decade of TAPSOFT in 1995, Ehrig and Mahr, after admitting some disproportion between the original claims of formal methods and their real impact on software practices, were however insisting on the need of rooting engineering practices on "the contributions from theoretical and conceptual work". That call was taken up and expanded by the authors first in a talk at the last TAPSOFT (Lille,1997) [2, 3] and later on in some papers advocating the use of "well-founded methods" more than "formal methods" (see [5] for a general presentation). Well-founded methods are precisely rooted on theoretical and conceptual models, but presented in a way that is friendly for the user and more concerned with the practical engineering needs. In this paper, addressing the classical problem of modelling the behaviour of a system, we present in a sense a paradigmatic journey from purely formal and textual techniques to derived visual notations, with a further attention first to code generation and finally to the incorporation into a standard practical notation such as the UML [13]. A bit more precisely, we show how starting from

H.-J. Kreowski et al. (Eds.): Formal Methods (Ehrig Festschrift), LNCS 3393, pp. 167–189, 2005.

a formal specification technique, namely, positive conditional specifications with initial semantics of labelled transition systems, expressed using the CASL specification language [6, 12], we can devise a new visual paradigm, which can also be adopted for an extension of UML. The paradigm is centered on the interaction charts, which are diagrams able to express also a proactive/autonomous behaviour, in opposition to the only reactive behaviour of the state charts and of the UML state machines.

The first main new contribution of this paper is the introduction of the executable interaction charts, by which we try to tackle the problem of the treatment of the nondeterministic choice among various alternatives when moving from abstract formal notations to more practical notations that need a kind of operational/executable semantics in order to ease the passage to code. Whereas there are no needs to restrict the alternatives in the first case, namely, it is possible specify a system that may nonderministically choose among a set of activities of any kind (internal, inputting, outputting, a mixture of inputting and outputting), in the latter either the sets of activities among which to choose are restricted (for example, only inputting and at most one internal, as in UML and Ada programming language) or some mechanism is introduced to be able to discover which alternatives are feasible in a certain situation (e.g., event queues/pools of UML). Executable interaction charts follow the second choice, by proposing the use of abstract buffers. The result is an abstract and executable visual notation to specify/model interactive behaviour, a kind of behaviour commonly found in "client" components, proactive agents and so on.

The second contribution of the paper is the introduction of the interaction machines, which are essentially executable interaction charts in a notation that can be easily incorporated, as an extension, in the UML notation.

We start in Sect. 2 by briefly summarizing the use of conditional specifications for modelling the behaviour of systems, then we introduce in Sect. 3 the interaction charts, showing how they have been derived from the corresponding conditional specifications. In Sect. 4 we present the executable interaction charts, and finally in Sect. 5 we show how they can be used to extend UML with a new kind of diagrams, the interaction machines.

2 Free Positive Conditional Specifications for Modelling Behaviour

Here, we use the word *system* to denote a dynamic entity of whatever kind, and so evolving along the time, without any assumption about other aspects; thus a system may be a communicating/nondeterministic/sequential/... process, a reactive/parallel/concurrent /distributed/... system, but also an object-oriented system (a community of interacting objects), and an agent or an agent system.

For modelling the behaviour of systems we adopt the well-known and accepted technique based on labelled transition systems (see [11, 16, 4]), which is today standard, widely used, and proven adequate in many cases, and there is a huge literature. For example, labelled transition systems are the basic formal