Softw Syst Model (2005) 00: 1–2 / Digital Object Identifier (DOI) 10.1007/s10270-005-0082-5

Special section on St.Eve workshop

Introductory paper

Tommaso Bolognesi[1], **John Derrick**[2]

[1] CNR, Istituto di Elaborazione della Informazione, Istituto die Scienza e tecnologie dell´Informazione "A. Faedo", Area della Ricerca CNR, via G. Moruzzi, 1, 56124 Pisa, Italy; e-mail: t.bolognesi@iei.pi.cnr.it
[2] University of Sheffield, Department of Computer Science, Regent Court, 211 Portobello St., Sheffield S1 4DP, UK; e-mail: J.Derrick@dcs.shef.ac.uk

Published online: ■■ 2005

Specifications of complex systems are usually based on either states or actions (events). Some people believe that the first step in system development should be the identification of the right global state structure. Other people start characterizing the system by describing its interactions with the environment, that is, by identifying event patterns (e.g. use cases), without worrying, at least initially, about the shape of the state. Formal specification approaches such as Abstract State Machines, B, CSP, LOTOS, Predicate/Transition Nets, Statecharts, TLA, Z, imply a considerable bias towards one of these two ways of conceiving system behaviour.

Abadi and Lamport [1] observe that, while the two approaches are, in some sense, equivalent ('an action can be modeled as a state change, and a state can be modeled as an equivalence class of sequences of actions'), they have taken, in the past, different formal directions. State-based approaches are often rooted in logic, while event-based approaches have algebraic roots. These differences tend to obscure the real differences and possible bridges between state-based and event-based intuitive thinking, and obstruct the identification of solid criteria for choosing among specification methods. Often this choice is driven by political considerations, or even by dogma, more than by genuine technical insights.

In a time in which formal methods have begun to show their effectiveness in addressing (some aspects of) large-scale, industrial level system development, it seems convenient to distill fundamental similarities and differences among approaches, to promote convergence, and to stress paradigms over individual languages, thus providing a scenario that improves awareness and effectiveness of choices.

This was the background to the St.Eve Workshop – State-oriented vs. Event-oriented Thinking in Requirements Analysis, Formal Specification and Software Engineering, which was held as a satellite event of the 12th International FME Symposium, Pisa, Sept. 13, 2003. The purpose of the workshop was to stimulate discussion on these topics; in particular, participants have been confronted with the following questions:

- What is the shape of your mental landscape, when you start conceiving a complex (concurrent, reactive, distributed) system? Is it a structure of state variables (relations, functions), or a pattern of events in time?
- Is the choice between a state-oriented and an event-oriented approach dependent on the type of system to be described? How?
- How does a system description in natural language affect the choice between state-oriented and event-oriented formalisation? How does the requirements analysis process affect the choice?
- The two approaches don't have to be mutually exlusive. Is it easy/desirable to move from one to the other? At which stage of development would one do that?
- Can one integrate the two approaches, keeping their individual advantages? If so, can one formally refine a purely state-oriented or purely event-oriented description into a hybrid one?

As suggested by some of the questions above, our aim was to specifically extend our investigations to the very early stages of system conception, including the 'preformal' brainstorming phase during which ideas about system behaviours pop up and are collected in a rather free, unstructured manner. In this respect, we were potentially open to interdisciplinary contributions, e.g. from areas such as cognitive psychology and natural language processing.

The Workshop consisted of eleven papers and attracted a wide and varied audience. Following the workshop three papers have been selected, and revised for publication in this issue.

MS ID: SoSyM0082
19 April 2005 13:45 CET

It was interesting to observe that the papers in the workshop (and thus those selected here) concentrated more on the specification and design phase as opposed to the the pre-formal, intuitive phase of system conception. This is perhaps reflective of the areas of work currently being concentrated on, and it would be nice to see how, in the future, this work shapes requirements analysis if indeed it does at all.

On the other hand, all three papers strongly emphasize on the comparison and the possibility to integrate the best features and advantages of the two paradigms, that are respectively represented by CSP an EB3 (action-based languages), and by B an Z (state-based languages). Although we are still left with some curiosity about possible arguments from cognitive psychology or linguistics that might shed light on the essentials of state-oriented vs. event-oriented thinking, we welcome all three papers in the present special section for the insights they offer on effective combinations of key expressive features from both paradigms.

The paper by Fraikin, Frappier and Laleau presents a detailed comparison of specifications in the two paradigms, using the well known B method, and the EB3 process algebra, and refers in particular to the development of Information Systems. The comparison is conducted under four criteria: expressiveness in behavioural specification, validation against user requirements, formal verification, and flexibility to change and evolution.

The subsequent paper, by Evans and Treharne, takes a further step and presents an integration of CSP and B. To quote, the aim of the paper is to demonstrate that it is possible to integrate two well established formal methods whilst maintaining their individual advantages. By individual advantages the authors mean that the languages, their semantics, and their tool-support can be used unchanged, and thus it is in some sense a compositional approach to integration. The integration depends on consistency checks between the CSP and B parts of a combined specification, and the authors argue that these checks can themselves be undertaken in a compositional fashion.

The final paper in this section also looks at combining established specification languages, this time integrating Z, CSP and the refinement calculus. The resultant language is called Circus by the authors. In this paper Cavalcanti, Sampaio and Woodcock extend Circus by adding a third important expressive dimension, namely object-orientation, with the notion of class. Their specific objective is to improve the facilities for the description of complex data structures and have a target programming language in mind, namely an extension of Java with CSP constructs. The paper thus presents the syntax, semantics of this OhCircus language and discusses refinement in the language. The essential difference between the two approaches to integration in these two papers is that in the latter state transitions and events are decoupled, this allows for a potentially richer set of implementation platforms and languages, although at the expense of simplicity of integration.

We hope that the present selection of papers from the St.Eve Workshop will trigger in the reader further interest on the integration of formal specification methods, on the investigation of specification paradigms over individual notations, and on the establishment of powerful specification languages and methods able to reflect as transparently as possible, within a formal setting, the multifaceted ways in which we intuitively conceive and describe complex behaviours.

References

1. Abadi M, Lamport L (1993) Composing Specifications. ACM Transactions on Programming Languages and Systems 15(1):73–132, January 1993