Computing and Informatics, Vol. 22, 2003, 1–2, V 2003-Oct-2

FOREWORD

Dines Bjørner

Department of Computer Science and Engineering
Institute of Informatics and Mathematical Modelling
The Technical University of Denmark
DK–2800 Kgs.Lyngby
Denmark
e-mail: db@imm.dtu.dk
www: http://www.imm.dtu.dk/~db

Before software can be designed, its requirements must be understood. And before computing systems requirements can be seriously expressed, the application domain must be reasonably well understood.

Aeronautics engineers, in the more serious parts of their work, rely on precise models of aerodynamics. Aerospace engineers, likewise, rely on celestial mechanics. Mechanical engineers rely on Newton's laws. Electronics engineers rely on laws of electricity, plasma physics, etc.

But, for some strange reason, software engineers hardly, if ever, refer to any theories about the domains for which they program applications.

This lack of professionalism amongst software engineers cannot prevail. As mathematically precise languages and proof systems for the formal specification — and proofs of properties of models — of domains, requirements and software designs evolve, as industrial–strength tools for creating and analysing such specifications for seemingly large–scale applications emerge, and as legal responsibility for software failures take hold, we shall see our field becoming more professional.

We shall see that trustworthy software (whose development needs several software engineers) will be based on reasonably formally described domains — and that theories will be researched and developed for such domain models. That software will be based on reasonably formally described requirements, larger parts of which will relate formally to the domain models. And that the ensuing software will be likewise formally, first abstractly, later more concretely described. The abstract software designs will be formally related to the requirements. Many design properties will be proved correct with respect to the requirements — in the context of assumptions about the domain as expressed (already) in the domain models.

This double issue of Computing and Informatics features seven[1] papers that describe corresponding formal specification languages: Aspects of their semantics and logic. These languages, including those mentioned in the footnote, together with their proof systems and tool sets, are of the kind that is today helping the field of software engineering to become professional.

One can roughly "classify" the languages covered in this ussue and its subsequent issues, in three ways: There are the model oriented specification languages (ASM, B (eventB), RSL, VDM–SL, and Z), the property oriented languages (CafeOBJ, and CASL), and the languages which specifically cater for temporal properties (DC and TLA+).

But, as we see unfolding today, languages like the above need be used in conjunction with other formalisms in order to believably cover a full spectrum of attributes of domains, requirements and software designs. Such "languages" as Petri Nets, Statecharts, Live Sequence Charts, and even the diagrammatic facets of *Entity Set Relations (ER)* (as for example presented in UML Class Diagrams) offer themselves. Their virtues are often that they provide diagrammatic means of illustrating crucial concepts. And, although: *"A formula is worth a thousand pictures. A picture is still worth a thousand words."* Intense and exciting research is today taking place unifying the underlying theories of these specification paradigms (ie., in integrating formal methods). *Diagrammatic reasoning* is likewise an emerging, exciting research topic.

For now, enjoy the present papers. They provide, in some issues of Computing and Informatics, a significant insight into possibly the most important (textual) specification languages.

I am most grateful to the many authors of the papers of this issue of Computing and Informatics to have answered my solicitation so vigorously and professionally. And I am very, very grateful to the many referees who have all worked very hard, providing exceptionally thorough review reports. Their names will appear in a later issue of this journal.

Holte, October 2, 2003

[1] A paper on the Duration Calculi (DC), authored by Michael Reichhardt Hansen, will be published in a next issue of this journal. A paper on VDM–SL (The Vienna Development Method Specification Language), authored by John Fitzgerald, will appear in a somewhat later issue of Computing and Informatics.