Future Trends of TAPSOFT

Hartmut Ehrig Bernd Mahr
Technische Universität Berlin
Franklinstraße 28/29, D-10587 Berlin
e-mail: {ehrig, mahr}@cs.tu-berlin.de
January 1997

Preface and Summary

The TAPSOFT-conferences on Theory and Practice of Software Development started 1985 in Berlin and were held bi-annually in Pisa, Barcelona, Brighton, Paris, Aarhus. In 1995 it was decided to combine TAPSOFT with ESOP (European Symposium on Programming) to the new conference ETAPS (European Joint Conference on Theory and Practice of Software) which will start 1998 in Lisbon. For this reason TAPSOFT'97 in Lille is the last TAPSOFT in the old style combining the conferences CAAP (Colloquium on Trees in Automata and Programming) and FASE (Formal Aspects of Software Engineering).

During FME'96 (Formal Methods Europe) in Oxford it was decided to create a new European Association, called EASDS (European Association of Software Development Science), which will cooperate with EATCS (European Association on Theoretical Computer Science) on the theoretical issues of software science, but should especially care about software development from the practical point of view.

Motivated by these events and an invitation of the first author to the panel of TAPSOFT'97 on "Theoretical Computer Science and Software Science: The Past, the Present and the Future" the first author asked a number of colleagues from different countries concerning their opinion on "Future Trends of Theoretical Computer Science and Software Development Science". After a careful analysis of their replies we have decided to summarize the discussion under the heading of the following four trends:

- From Diversity of Mathematical Concepts to Unification of Computational Models and Semantic Theories
- From Algebraic Specification to Integration of Formal Techniques
- From Trees to Graphs, Graph Transformations and Visual Languages
- From Abstract Data Types to Object-Oriented Techniques and Continuous Software Engineering

The subsequent more detailed discussion of these trends expresses our own opinion and is also meant as a basis for the TAPSOFT'97 panel discussion.

Acknowledgement

For stimulating contributions concerning future trends we would like to thank Michael Arbib, Ed Blum, Heiko Dörr, Gregor Engels, Gerhard Goos, Klaus Grimm, Tony Hoare, Stefan Jähnichen, Hans-Jörg Kreowski, Michael Löwe, Jose Meseguer, Ugo Montanari, Fernando Orejas, Grzegorz Rozenberg, Horst Reichel, Herbert Weber and Emo Welzl. Finally, we are grateful to Maike Gajewsky for structuring the material and final layout.

1 From Diversity of Mathematical Concepts to Unification of Computational Models and Semantic Theories

One of the aims of Theoretical Computer Science is to present suitable mathematical models and solutions for different concepts and problems in Practical Computer Science. In the case of programming paradigms, like functional, procedural, logical or parallel programming, this leads to a variety of different computational models. Moreover, there are different styles of semantic theories, like operational, algebraic, denotational or axiomatic, for specification and programming and a diversity of formal reasoning tools. This kind of diversity of mathematical concepts is similar for formal modelling in all areas of Practical Computer Science. Numerous interesting results have been published in the literature. For practical applications, however, especially in an industrial environment, the diversity of specification and programming languages with their different mathematical models is problematic. For practical use also suitable methodologies and tools supporting the software development process are missing. It is pointed out by Heiko Dörr and Klaus Grimm from Daimler-Benz that automation based on formal methods should be pushed for correctness and cost reasons, but there should be a limitation to a small number approaches and an effort on the development of suitable methodologies and tools.

In fact, in practice there is a strong concentration on those specification and programming languages, which are supported by efficient and reliable commercial tools. Since this tendency is also shared by most of the European funding agencies this has become a severe problem for basic research. Certainly we need, as proposed by Gerhard Goos and others, a better transfer of results from theory to practice. But in order to transfer interesting new results it is pointed out by Emo Welzl, that theoreticians must be able to continue with research, which is impossible without further funding for basic research. On the other hand the diversity of existing mathematical concepts for computational models, semantic theories and formal reasoning tools should be unified within Theoretical Computer Science. This kind of unification on the basis of abstract general models is considered an important future trend by Tony Hoare and Ugo Montanari, in order to create a better consolidation of theory and successful transfer of technology.

2 From Algebraic Specification to Integration of Formal Techniques

Algebraic specification techniques have been very successful for the specification and reasoning about abstract data types, functional and logic programming, as well as concepts for structuring and refinement of software systems. Several useful algebraic specification languages have been developed, implemented, and applied in numerous research projects. However, due to the diversity of different concepts and styles and the lack of commercial tools, there are only few applications in truely practical environments up to now. This unsatisfactory situation will certainly improve once the new algebraic specification language CASL developed by the common framework initiative for algebraic specification (CoFI) is available. On the other hand there are already several other multi-purpose specification languages, like VDM and Z, and specific techniques and languages for concurrent, distributed, embedded and reactive systems, like Petri nets, CSP, CCS, process algebras, statecharts, and temporal logic. Moreover, there is a huge number of semi-formal specification techniques, like entity-relationship diagrams and object-oriented design techniques, which are used for software development. Unfortunately, different techniques are used in different phases of software development, like requirement and design, and for different aspects of the system, like static and dynamic views.

For Theoretical Computer Science it is proposed by Jose Meseguer, Fernando Orejas, Gregor Engels and others, that interoperability and integration of formal techniques is an important future trend. For Software Development Science it is important to allow heterogenity, to have concepts for integration (Gregor Engels, Stefan Jähnichen, Herbert Weber) and language independent concepts for specification and programming in the large (Fernando Orejas, Gerhard Goos). Modularity, well-studied for algebraic specification languages, is a must for all kinds of languages, even for new kinds of languages, like the neural simulation language NSL (Michael Arbib), and must also be extended to integrated specification techniques, like algebraic high-level Petri nets (Herbert Weber).

3 From Trees to Graphs, Graph Transformations and Visual Languages

The notion of trees is still an important concept in Computer Science and has been studied in detail especially within the CAAP-conferences for more than 25 years by now. Although the history of graphs and graph grammars in Computer Science also started in the late 60'ies and early 70'ies the importance of these concepts for Software Science and Development has been recognized by the scientific community only recently. Today, graphical user interfaces as well as graphical visualization and animation techniques are standard due to increasingly high performance and capacity of workstations and PC's. On the other hand the concepts behind these graphical techniques, which are important for

the development of visual languages, are not yet well understood and studied by computer scientists. Graph grammars and transformations have been investigated by a small international community up to now, including Azriel Rosenfeld, Grzegorz Rozenberg, Hans Jürgen Schneider, Hans-Jörg Kreowski, Manfred Nagl, Ugo Montanari, Andrea Corradini, Gregor Engels, Jean Claude Raoult, Bruno Courcelle and several people in Berlin, but the importance as a future trend is also pointed out by scientists in Software Engineering like Herbert Weber and Gerhard Goos. The importance of rewriting and rule-based concepts for all kinds of structures is apparent for numerous aspects of specification, reasoning and programming in software science, communication technology and artificial intelligence. Especially rewrite logic (Jose Meseguer) and high-level replacement systems (Ugo Montanari), a generalization of graph transformations to high-level structures in suitable categories, seem to be important future trends supporting the development of formal interoperation of different specification techniques (Jose Meseguer) and visual languages (Gregor Engels, Hans-Jörg Kreowski).

4 From Abstract Data Types to Object-Oriented Techniques and Continuous Software Engineering

The notion of abstract data types, developed and formalized already in the 70'ies, is still one of the most important concepts in Computer Science, especially in Theoretical Computer and Software Development Science. In fact, abstract data types have been extended by various parameterization, transformation and modularization concepts which are most important for horizontal and vertical structuring of software systems. More recently abstract data types, mainly used to model static aspects, have been extended by state-oriented and dynamic aspects, leading to the concept of dynamic algebras and dynamic abstract data types. On the other hand the object-oriented paradigm has turned out to be one of the most important concepts for architectural design and programming of all kinds of software systems, especially supported by the commercial success of C++. In fact, there are several formal concepts, like classical and dynamic abstract types, process algebras, co-algebras, actor systems and attributed graph transformations, which have the capability of modelling certain aspects of object-oriented techniques. But it is still open and considered as an important future trend by Horst Reichel, Gerhard Goos and others, to develop a widely accepted formal model for the object-oriented design and programming paradigm.

Another important aspect in the area of software engineering, database and information systems as well as communication technology and computer networks is the problem of continuous change of requirements for already existing software systems in all areas of administration, commercial services and industry. This means that today maintenance of software includes re-engineering and hence continuous software engineering. Although this problem is known and faced in practice since the very beginning it has become a matter of research only recently. Unfortunately, formal methods for software development have been almost ne-

glected the area of re-engineering up to now. An important problem is that the adaption of the software means to change the design patterns online, because shut down of the system is highly undesirable for commercial reasons. Being involved with these problems in practice it is proposed by Herbert Weber and Michael Löwe that research concerning continuous software engineering and - even more general - for evolutionary systems in different areas of science is an important future trend. Most recently it has been shown by Michael Löwe (now managing director of a software institute, which is a daughter of an important insurance company) that among all existing semi-formal and formal methods the theory of graph transformation has the greatest potential to solve the problems of re-engineering and continuous software engineering mentioned above.