Int J Softw Tools Technol Transfer (2003) 4: 401–404 / Digital Object Identifier (DOI) 10.1007/s10009-003-0108-7

Special section on ASTEC: an experience in the establishment of collaboration between academia and industry

Preface by the section editors

Bengt Jonsson[1], **Konstantinos Sagonas**[2]

[1] Department of Computer Systems, Uppsala University, Sweden; e-mail: bengt@docs.uu.se
[2] Computing Science Department, Uppsala University, Sweden; e-mail: kostis@csd.uu.se

Published online: 12 June 2003 – © Springer-Verlag 2003

Abstract. ASTEC (Advanced Software TEChnology) is a competence center for industrially relevant research on software technology, centered in Uppsala, Stockholm, and Västerås. It is organized as a consortium between a group of Swedish companies and academic institutions, supported and partially funded by VINNOVA, the Swedish Agency for Innovation Systems. In this introduction, we outline the main ideas behind the creation of ASTEC, its activities, and some of the experiences that we have gained from running the center. This issue also contains a set of companion articles, which describe some of the main projects and results obtained in the context of ASTEC.

Keywords: Competence centers – Software engineering – Embedded systems – Academia-industry collaboration

1 Introduction

This collection of articles presents work conducted within ASTEC (Advanced Software TEChnology), a Swedish *competence center* which concentrates on tools and techniques for software development.[1] Since 1995, a total of 29 competence centers, intended as focussed (both research-wise and geographically) environments for collaboration between academia and industry in various research areas, ranging from forestry to high-speed electronics, have been supported by the Swedish National Board for Industrial and Technical Development (NUTEK), and (from 2001) the Swedish Agency for Innovation Systems (VINNOVA).[2] NUTEK and VINNOVA have, over the last decade, supported a number of national research activities in strategic and industrially relevant research areas. ASTEC was formed in 1995 and is the only one of these competence centers that focuses exclusively software technology.

Industrial and academic collaboration should be particularly relevant for research on software systems: many advances in software technology occur in universities but on the other hand the major part of software production occurs in industry. Such a collaboration thus offers a unique opportunity for industry to exploit recent advances in software technology and for researchers in academia to evaluate the effectiveness of new techniques in a "real-world" environment.

Though this observation is probably universal, it is particularly applicable to a country like Sweden. Several major products of Swedish industry, e.g., data communication and process control systems, are to a significant extent based on software. Swedish academia has a strong tradition of research in the areas of formal methods, functional and logic programming, real-time and embeddded systems which have potential applications to software development. The indirect impact of this research on industrial practice, through mobility of individuals and ideas, has been noticeable: functional programming and formal methods have influenced software development in the Swedish telecommunications industry, as witnessed, e.g., by the creation of the languages Erlang [1], SDL [3], and TTCN [6]. ASTEC is intended to build upon this tradition, and strengthen direct contacts between academic research and industrial practice, so that advances in academic research can indeed benefit industrial software development, which in turn can guide the agenda for academic research.

ASTEC is a consortium of academic and industrial partners with a strong interest in research on industrially applicable software technology. The actual partners are:

[1] See also `www.astec.uu.se`.
[2] In 2001, NUTEK ceased to exist and some of its responsibilities were transferred to the newly formed VINNOVA

- research groups at *Uppsala University*, the *Royal Institute of Technology*, the *Swedish Institute of Computer Science* (SICS), and *Mälardalen University*, working mainly on formal methods, functional, logic and constraint programming, compilation, and on embedded, distributed, and real-time systems, and
- companies with a substantial software production and thus a large interest in software development: *ABB*, *CC-Systems AB*, *Ericsson*, *Mecel AB*, *Validation AB*, and *Volvo Technical Development Corp.*, and companies that produce tools for software development: *IAR Systems AB*, *OSE Systems AB*, *Prover Technology AB*, *Telelogic AB*, *UPAAL AB*, *Virtutech AB*, and *Volcano Technologies AB*[3].

A substantial body of financial support is provided by VINNOVA. The industrial and academic partners also contribute significant resources, so that VINNOVA, the industries, and academia each contribute around one third of the total resources to the competence center.

2 Research program

The purpose of ASTEC is to conduct research on industrially applicable techniques for software specification, design, and implementation. As a basis for the research activities, the ASTEC consortium has developed a *strategic research plan*, which sets up long-term research goals. The overall vision of the research plan is to contribute in developing high-level specification and programming techniques, together with powerful automatic tools, which can assist in producing high-quality software for complex applications with less effort. The technical challenges have been structured into the following three areas:

- **Verification and validation** is concerned with high-level notations for expressing requirements and design specifications, together with tools and (formal) methods for analysis of specifications for the purposes of verification, validation, test generation, and tracing of requirements.
- **Programming language implementation and compilation** is concerned with the implementation and use of high-level (concurrent) programming languages, together with the development of compilation technology for (time- or space-) efficient program execution and code generation for different architectures.
- **Real-time, embedded, and distributed systems** is concerned with features specific to software development for real-time, embedded, or distributed systems, such as predictability, timeliness, scheduling, and distribution.

In short, ASTEC focuses on techniques for software in which aspects of automatic control and communication are dominant. Thus, two application areas within the scope of ASTEC are:

- **Automotive and vehicle control systems**, including embedded and safety-critical software, often deployed on a distributed network, and
- **Data- and telecommunication systems**, with requirements on mobility, high distribution, massive concurrency, and code replacement without disruption of the continuous operation of the system.

Within the scope of this plan, research is conducted aiming both at being of high scientific quality and at the same time being industrially relevant. Some typical motivations for particular research projects are to develop and/or apply a new technique to an industrial problem, or to develop further some technique already in use, maybe to the point where it can be applied in industrial practice or in commercial software tools.

There are several advantages to forming a consortium and formulating a strategic research plan, rather than just running individual collaboration projects on an ad hoc, case-by-case basis. A consortium, like the one of a competence center, provides infrastructure for continuous collaboration in the form of long-term goals, personal contacts, agreed-upon rules for ownership and exploitation of research results. The administrative overhead needed to start a new project is minimal, since personal contacts and rules are already established.

3 Experiences

This special issue illustrates aspects of the activities that have been conducted under the aegis of ASTEC mainly during the period 1997–2001. Some of these projects are still ongoing. Several ASTEC projects are not represented in the following collection of articles, partly because of space restrictions, and partly because some projects have been less successful than others. Trying to consider why certain projects have yielded more interesting results, we will in this section attempt to generalize slightly and elaborate on some of the factors that we believe have been important for the success of a long-term collaboration project between academia and industry.

From the perspective of academia, some industrial problems may appear rather easy at first sight. If a superficial solution to an abstraction of a problem is proposed, one may later discover that the proposed technique did not yield the expected benefits because the original problem contained many difficulties stemming, e.g., from a variety of environmental constraints. For instance, formal specification and modeling may seem to improve the software development process for many types of applications. However, without support from tools that assist in activities like debugging, code generation, documentation, test generation, confidence building, etc. the gains may be rather limited, and may not outweigh

[3] "AB" is the Swedish equivalent of "Inc.".

the cost of formalization. Our experience is that formal methods can bring benefits when combined with developing an industrial-strength tool that supports these methods. As another example, certain compiler optimizations may yield substantial performance improvements on synthetic benchmarks, but in practice one often discovers that the actual performance in big applications is a result of many factors, such as memory management, calling functions from libraries that are outside the reach of the optimizations, cache behavior when the application contains a variety of components, etc. A general lesson learned is that industrially relevant research should take these factors into account when drawing conclusions.

Related to the above, a notable advantage of the competence center framework is that it offers access to production-size industrial software products which can be used as case studies. As mentioned, the effects of optimizations on small programs are often different from those observed on entire software systems. Evaluating compilation techniques on production-size software besides revealing the obstacles that must be solved in order to integrate a technique into a commercial tool, often reveals new problems that can become the topic for further research. As another example, when attempting to formally model and verify industrial-size software products, it appears that the notion of "correct software" becomes very elusive, to the point where one may come to the conclusion that this notion is meaningless: in most cases, it is impractical to define a complete set of requirements. The interesting question then becomes how to best integrate formal verification technology into the development and maintenance of software.

Finally, long-term research collaborations between industry and academia require personal contacts and commitment from both sides. A situation that in our experience works well, is when some person is affiliated with both partners. For example, some Ph.D. students within the context of ASTEC are or have been half-time employed by one of the participating companies. The majority of their time within the company is spent on work related to their Ph.D. research, partly on research itself, and partly on integrating the results into the company's products and on internal development. Our experiences from these students is positive, but it is important that the topic of the doctoral research be supported within the framework of a collaboration between academia and the company.

4 About this issue

Having briefly described our experiences, we overview the articles in this issue grouping them based on the three main research areas of ASTEC. The boundary between these areas and the corresponding activities described is not always clear cut.

4.1 Verification and validation

The first article by Fredlund, Gurov, Noll, Dam, Arts, and Chugunov [2] presents a tool for verifying software written in Erlang. Erlang [1] is a concurrent functional programming language designed to ease the development of large-scale distributed control applications. So far, it has been used quite successfully in the telecommunication industry, both within Ericsson Telecom, where it was designed and developed, and by other companies. Essentially, the verification approach considered in the article consists in proving, with a Gentzen-style proof system, that Erlang code satisfies a set of properties formalized in Park's μ-calculus [9] extended with Erlang-specific features. The article reviews the mathematical machinery, motivates the chosen framework and discusses reasoning principles, such as inductive and compositional reasoning, which are essential for successful verification of typical software written in Erlang. The main design objectives of the verification tool, a proof assistant, are to achieve a satisfactory degree of automation, proof reuse, easy navigation through proof tableaux, and meaningful feedback about the current proof state, so as to require user intervention only when this is really necessary, and to assist its user in taking informed proof decisions. The experiences of applying the verification tool in an industrial case study are summarized in the paper and in a concluding section an approach for supporting verification in the presence of program libraries is outlined, to permit verification of modular Erlang software. The results reported in the paper is the outcome of a joint project between the Formal Design Techniques group at the SICS and Ericsson's Computer Science Laboratory.

4.2 Programming language implementation and compilation

The work on the concurrent functional programming language Erlang within the context of ASTEC is not restricted to the verification of its applications. In [7], Johansson, Pettersson, Sagonas, and Lindgren describe their experiences from embarking on a multi man-year project, called HiPE (High Performance Erlang), aiming to improve the performance aspects of publicly available Erlang implementations. Until recently, Erlang implementations were based on emulators of virtual machines and thus were slow even compared to implementations of other functional languages. An outcome of the HiPE project is the freely available homonymous system which is an efficient just-in-time native code compiler for Erlang. The article describes the different phases of HiPE's development, the main compilation techniques used in HiPE, and reports on HiPE's performance both on benchmarks and on large-scale industrial applications of Erlang. In addition, the authors critically examine the design decisions taken and report on their experiences from participating in an academic project trying to keep up with the concurrent development the open-source Erlang system – upon which HiPE is based – by the research department of Ericsson. Since October 2001, HiPE is integrated in the open source and commercial versions of the Erlang/OTP system from Ericsson.

4.3 Real-time embedded and distributed systems

In the next article, Engblom, Ermedahl, Sjödin, Gustafsson, and Hansson describe their experiences in developing *worst-case execution-time* (WCET) analyses for embedded real-time programs [4]. The purpose of WCET analysis is to provide information about the worst possible execution time of a piece of code before using it in an actual system. In real-time embedded systems (e.g., in embedded microcontrollers), *safe* (and preferably *tight*) WCET estimates are used to e.g., perform scheduling, to determine whether performance goals are met for periodic tasks, and to check that interrupts get serviced in sufficiently short reaction times. The article presents a modular architecture for an WCET tool and describes how control-flow graphs of programs should be represented and analyzed, how pipeline and cache behavior and timing should be modeled, how the components of the tool can be validated, and how the WCET analysis can be integrated in an industrial development environment. This work has been performed in close cooperation with the embedded systems programming tools manufacturer IAR Systems AB.

Finally, in [5], Håkansson, Jonsson, and Lundqvist present a technique for automatically generating on-line test oracles from specifications for embedded systems written in temporal logic. Since embedded systems are increasingly employed in safety-critical applications such as in cars and airplanes, the *validation* that various safety requirements are met by their specification and their implementation becomes necessary. As it is tedious to perform a large number of tests manually, testing that the system conforms to its safety requirements should be automated so that a wide range of possible input values is covered. Also, ideally, test executions that already revealed problems should be automatically filtered out. The article describes how a restricted subset of TRIO [8], an expressive first-order logic with special constructs for handling metric (linear) time, can be used in a syntax-directed way to express quantitative properties of durations and properties of time-dependent behavior. It is argued that this approach presents a lesser effort at integrating the use of temporal logic as a language for expressing requirements into the process of developing embedded software. This automatic generation of test oracles has been implemented and exercised in two case-studies at Volvo Technical Development Corp., on a cruise control and on a throttle module, and the article reports on these experiments.

References

1. Armstrong J, Virding R, Wikström C, Williams M (1996) Concurrent Programming in Erlang. Prentice Hall
2. Fredlund L-Å, Gurov D, Noll T, Dam M, Arts T, Chugunov G (2003) A verification tool for Erlang. Int J Softw Tools Technol Transfer 4(4): 404–419
3. Belina F, Hogrefer D (1991) SDL with applications to protocol specification. Prentice Hall
4. Engblom J, Ermedahl A, Sjödin M, Gustafsson J, Hansson H (2003) Worst-case execution-time analysis for embedded real-time systems. Int J Softw Tools Technol Transfer 4(4): 436–454
5. Håkansson J, Jonsson B, Lundqvist O (2003) Generating on-line test oracles from temporal logic specifications. Int J Softw Tools Technol Transfer 4(4): 455–470
6. ISO/IEC 9646-3 (1997) Information Technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 3: The Tree and Tabular Combined Notation (TTCN). Available at `ftp://ftp.npl.co.uk/pub/ttcn/`
7. Johansson E, Pettersson M, Sagonas K, Lindgren T (2003) The development of the HiPE system: Design and experience report. Int J Softw Tools Technol Transfer 4(4): 420–435
8. Morzenti A, Mandrioli D, Ghezzi C (1992) A model-parametric real-time logic. ACM TOPLAS, 14(4): 521–573, Oct 1992
9. Park D (1976) Finiteness is mu-ineffable. Theor Comput Sci 3: 173–181