1999

Software Engineering to Our Planning Horizon

Luqi

Elsevier

Luqi, "Software Engineering to Our Planning Horizon", Electronic Notes in Theoretical Computer Science, Elsevier Science, 1999, Vol. 25, pp. 145-146.
https://hdl.handle.net/10945/65705

Electronic Notes in Theoretical Computer Science 25 (1999)
URL: http://www.elsevier.nl/locate/entcs/volume25.html 3 pages

Software Engineering to our Planning Horizon*

Luqi

Computer Science Department
Naval Postgraduate School
Monterey, CA 93943, USA

Manfred Broy

Institut Fur Informatik
Technishe Universitat Munchen
GERMANY

The Army Research Office, National Science Foundation, Office of Naval Research, and the Defense Advanced Research Projects Agency sponsored the 1998 Monterey Workshop on Engineering Automation for Computer Based Systems.

This workshop is the 6th in a series of international workshops with the general theme of increasing the practical impact of formal methods for software and systems engineering. The workshop took place in Carmel, California late 1998, hosted by the Naval Postgraduate School.

Since 1990, the previous workshops in the series focused on real-time and concurrent systems, software merging and slicing, software evolution, software architecture, and requirements targeting software. This workshop focused on engineering automation.

The objectives of the workshops are to encourage interaction between the research and engineering communities, exchange recent results, assess their significance and encourage transfer of relevant results to practice, communicate current problems in engineering practice to researchers, and help focus future research on directions that address pressing practical needs.

Over the past years, we have witnessed a slow but steady decrease in the gap between the theoretical and practical sides of the software engineering community. We hope that this trend will continue and will accelerate improvements in the state of software engineering practice and theory. Software problems have been quite visible to the public due to spectacular disasters in space missions or telephone black outs and are receiving increasing attention

* This research was supported by ARO(MIPR8GNPSAR042), NSF(CCR-9813820), ONR(N0001499WR20019), SPAWAR(N6600198WR00438).

with the nearing Y2K deadline. It is a good time to demonstrate concrete improvements in our discipline.

The continued doubling of computing speed and memory capacity every 18 months implies that the only constancy for large distributed systems, technology, tactics and doctrine may well be the idea that change is always inevitable. The dynamic aspect of systems is not supported by current practice and is seldom emphasized in current research. Software evolution research is extremely important for achieving modifiable and dependable systems in the future. Improved methods for reengineering are also needed to bring legacy systems to the condition where they can benefit from improvements in software evolution technology.

Thirty years ago, when the term software engineering was coined, there was lack of theoretical foundation for many practical concepts in computing. That is no longer true. A solid body of foundational work is available now that addresses many challenging issues related to software and computing, including specification techniques for systems and data, logical calculi for concurrent, distributed, and real-time systems, logical concepts related to interactive systems, and formal models of programming language semantics with a variety of inference systems.

The challenge is to put these results to work, to develop theory that better supports engineering needs, and to improve practice. This will require cooperation and a concerted effort from both theoreticians and practitioners. We will need advances in education and improvements in theoretical approaches to meet the demand of practical engineering for computer software. To be attractive to practitioners, formal methods, mathematical foundations and automated engineering tools need to provide return on investment. These approaches must be cost effective to successfully compete with other development methods, and the benefits they provide in terms of software quality must have sufficient economic value to justify investment in them.

These goals require some uncomfortable changes in the research community. Mathematical elegance is not enough for the success of an engineering theory: applicability, tractability, and ease of understanding are often more important in practice than logical completeness or conceptual elegance of the principles that guarantee the soundness of the methods. We must carefully separate the application of mathematics to demonstrate the soundness of a formal software model or to construct automated tools for engineers from the formal models that will be used "by engineers as design representations".

The formal aspects of computing cannot be studied in isolation if we are to have practical impact. The different aspects of technical, educational, and management issues are so closely intertwined in software engineering practice that it is risky and ineffective to study and develop them in isolation if practical applicability is a prominent goal. This puts interdisciplinary requirements on researchers and lends importance to interactions between experts from different specialties, such as those promoted by this workshop.

We have collected some excellent papers for the workshop. These articles are written by internationally renowned contributors from both academia and industry that examine current best practices and propose strategies for improvement, as well as a summary of the high points of the discussions at the workshop.

The broadest range of expert opinion and views were represented. Members of the academic, government, military and commercial world came to share their vision, insight and concerns. By synthesizing the expertise of these communities we hope to gain significant insight into the problems and solutions. The discussions ranged beyond the narrow confines of software and mathematics, to address engineering of systems containing hardware and people as well as software, and related issues that include requirements elicitation, management, and engineering education. Discussions at the workshop addressed technical advances in mature areas, such as a new decision procedure for a queue data type and novel types of model checking, as well as ideas for new directions, such as lightweight inference and co-algebraic models for interactive systems. The workshop helped to reduce the gap between theory and practice, and to recharge the research community to address problems of immediate concern. Workshop attendees identified and discussed both the technologically dependent and technologically independent trends within the engineering automation of computer based systems for the near term and out to our planning horizon.

It is our pleasure to thank the workshop advisory, program and local arrangements committees, and the workshop sponsors, NSF, ONR, DARPA, and especially ARO, for their vision of a principled engineering solution for software and for their many-year tireless effort in supporting a series of workshops to bring everyone together.