Non-Technological Issues in Software Engineering

Panel Chair:

Marc I. Kellner
Software Engineering Institute
Carnegie Mellon University

Panelists:

Bill Curtis
Software Engineering Institute

Tom DeMarco
The Atlantic Systems Guild

Kouichi Kishida
Software Research Associates, Inc.

Maurice Schlumberger
Cap Gemini Innovation

Colin Tully
Independent Consultant

Panel Session Overview

Marc I. Kellner

1. Introduction

Although much of the focus of software engineering has revolved around technological issues and concerns, it is also clear that there are a number of substantive non-technological problems that continue to be serious impediments to the more effective practice of software engineering. These non-technological issues include managerial, organizational, economic, political, legal, behavioral, psychological, and social factors. This panel session provides a forum for the discussion of such issues, among a group of internationally recognized authorities and the conference attendees.

It is obvious that technology will always play a vital role in software engineering. However, we must be attentive to the effective use of technology in improving the practice of software engineering. As pointed out by Curtis, Krasner, and Iscoe, "Software development tools and practices had disappointingly small effects in earlier studies, probably because they did not improve the most troublesome processes in software development." [3] Humphrey, Kitson, and Kasse report that "For low-maturity organizations, technical issues almost never appear at the top of key priority issue lists. This is not because technical issues are not important but simply because so many management problems must be handled first." [5] Brooks has concluded that there is no magical technological "silver bullet" to be found to solve the challenges of software engineering; instead he said, "The central question in how to improve the software art centers, as it always has, on people." [2] Boehm drew a similar conclusion after examining the cost driver factors in his COCOMO software cost estimation model: "Personnel attributes and human relations activities provide by far the largest source of opportunity for improving software productivity." [1]

---

[1] Author's address: Software Engineering Institute; Carnegie Mellon University; Pittsburgh, PA 15213-3890 USA; e-mail: mik@sei.cmu.edu

[2] This work was sponsored by the U.S. Department of Defense. The views and conclusions are those of the author and should not be interpreted as representing official policies, either expressed or implied, of the Software Engineering Institute, Carnegie Mellon University, the Department of Defense, or the U.S. Government.

While technology offers considerable potential for improvement, in many organizations the software process is sufficiently confused and incoherent that non-technological factors impede the effective application of technology [4]. In other words, there is considerable spinning of wheels; adding better technology may let those wheels spin faster, but that may dig a deeper rut rather than aiding forward progress.

Three important non-technological issues have been selected for discussion at this panel session, chosen from among a large number of identified issues. This set provides a reasonable degree of focus, while still allowing a broad array of considerations to surface:

1. The software engineering profession has not produced a cadre of capable/competent managers.
   - managerial, political
2. Software development is largely practiced as an individual creative activity, rather than a team effort.
   - behavioral, social, organizational, psychological, political
3. The software engineering community has not taken positive action to reduce the performance (e.g., productivity, quality) differences among individuals (or across teams).
   - behavioral, psychological, economic

These problems are more fully elucidated in Section 3 below. These issues are particularly crucial in software engineering because it is primarily concerned with large-scale software efforts. Successful large-scale projects generally require good management, effective group work, and highly skilled staff.

The panelists have each been asked to present their views on any two of these problems. Their comments are expected to address two important questions for each issue:

- What are the elements, facets, and consequences of the problem?
- What can/should be done to resolve the problem?

2. Panelists

Five eminent panelists have agreed to participate in this session. They offer a broad base of expertise in software issues, credible experience with practical software engineering projects, and international perspectives. Bill Curtis manages the Software Process Program at the Software Engineering Institute; he has published extensively in the software field, including two books on human factors in software engineering, and serves as an associate editor of several journals. Tom DeMarco was awarded the Warnier Prize in 1986 for "lifetime contributions to the information sciences"; DeMarco is also the author of four books on management and software engineering, including *Peopleware: Productive Projects and Teams* with Timothy Lister. Kouichi Kishida is a founder and the Technical Director of Systems Research Associates, Inc., in Japan; he has been very active in the technical community, serving as Program Co-Chair of ICSE-9, and helping found and lead the Japan UNIX Society and the Software Engineering Association. Maurice Schlumberger is Scientific Director for Cap Gemini Innovation, part of the largest software firm in Europe; he has managed several projects and organizations, and has served on the faculty of the University of California. Colin Tully is an independent consultant, based in the UK, with an international clientele; he has extensive industrial experience with many aspects of the software field, and has also been a member of the faculty of the University of York.

3. Issues

Numerous non-technological issues contribute to impeding the more effective practice of software engineering. Furthermore, these issues are highly intertwined and interdependent. For the purposes of this panel session, three problems have been selected which each serve to cluster several lower-level elements. The selection was based upon various considerations, including perceived importance, cohesion, and relative interest. However, these topics provide only a sampling of the important non-technological impediments. For example, legal and economic issues are largely absent from the three problems.

3.1. Poor Management

The software engineering profession has not produced a cadre of capable/competent managers. In many cases the criteria, climate, and rewards for good performance as a software engineer are at odds with the characteristics found in a good manager; consequently, people advancing up the technical ladder are often poorly prepared for management responsibilities. The negative impacts of this problem are profound. Boehm has stated that "Poor management can increase software costs more rapidly than any other factor." [1]

A number of difficulties can arise due to poor management, such as: unrealistic project plans due to poor planning, scheduling, and estimation skills; demotivation of staff due to inappropriate management styles for creative technical staff and lack of rewards for good performance; ineffective teams due to poor team building and other interpersonal skills; poor project

execution due to inadequate organization, delegation, and monitoring; technical difficulties due to lack of management discipline in areas such as configuration management, quality assurance, etc.; failure to invest in design, training, tools, methods, etc. due to a short-sighted rather than long-term perspective.

Appropriate actions to resolve this problem include: definition of dual career paths for technical and managerial staff with appropriate responsibility and compensation for each; training in managerial skills and techniques; active mentoring and supervision by senior managers; increased delegation of responsibility and matching authority.

3.2. Lack of Teamwork

Software development is largely practiced as an individual creative activity, rather than a team effort. The latter entails adoption of an engineering team vision of software development. Especially in the case of software engineering, which involves large systems and teams, it is the outcome of the project *team* that ultimately counts. Lack of teamwork can have serious consequences for cost, schedule, quality, and so forth. Nevertheless, software engineers often strive to retain the individual character of programming-in-the-small.

Several forces conspire to keep software work an individual activity, including: desire for autonomy; a culture that rewards individual efforts far more than team efforts; concentration of crucial application knowledge by a few individuals; desire for privacy regarding individual development efforts; the Not Invented Here syndrome and its more personal form (not invented by me); large productivity differences between individuals; political considerations of powerful individuals and of managers.

Various actions can be taken to improve this situation, such as: objective assessment of team contributions with appropriate rewards; development of an organizational culture that condones/rewards group efforts; active efforts to disperse crucial application knowledge across project staff; improvements in communication and coordination across organizational layers; adoption of egoless programming techniques.

3.3. Performance Differences

The software engineering community has not taken positive action to reduce the performance (e.g., productivity, quality) differences among individuals (or across teams). Boehm suggests that productivity ranges of 3:1 to 5:1 are typical, and some studies have documented ranges as high as 26:1 among experienced programmers. [1] Rather than tacitly accepting these differences, an organization can actually capitalize upon them to improve productivity. [1]

The observed variability is often related to misguided staffing practices, poor team development, inattention to the critical role of motivation, and poor management. Appropriate actions to increase the effective level of productivity include: enhanced education and training; investment in productivity aids (tools, methods); standard practices; enhanced motivation (professional development opportunities, recognition); effective staffing (top talent, job matching, career progression, team balance, phaseout) [1]; improved management.

4. Conclusion

In summary, it is clear that there are a number of non-technological factors acting as serious impediments to progress in improving the practice of software engineering. Much of the potential impact of technological advances may be blocked by these factors. In this session, five expert panelists offer a wealth of experience, insight, and international perspectives on three of these non-technological issues. The problems are seen to cause substantial negative impacts on software engineering, so their resolution offers tremendous opportunity. The panelists address both the problems and steps for their resolution.

Because these issues are outside the primary domain of expertise of many software engineering researchers and practitioners, it is especially important that they continue to be highlighted in ways that encourage participation and involvement. It is hoped that efforts such as this session will help promote a balanced approach to investigating and improving the practice of software engineering.

References

1. Boehm, Barry W.. *Software Engineering Economics.* Prentice-Hall, Inc., 1981.

2. Brooks, Fred P. "No Silver Bullet". *Computer 20*, 4 (April 1987), 10-19.

3. Curtis, Bill, Herb Krasner, and Neil Iscoe. "A Field Study of the Software Design Process for Large Systems". *Communications of the ACM 31*, 11 (November 1988), 1268-1287.

4. Humphrey, Watts S.. *Managing the Software Process.* Addison-Wesley, 1989.

5. Humphrey, Watts S., David H. Kitson, and Tim C. Kasse. The State of Software Engineering Practice: A Preliminary Report. Tech. Rept. CMU/SEI-89-TR-1, Software Engineering Institute; Carnegie Mellon University, February 1989.