Invited Industry Presentations (IIP)

François Coallier
Bell Canada
francois.coallier@bell.ca

Linda M. Northrop
Software Engineering Institute
Carnegie Mellon University
lmn@sei.cmu.edu

Dewayne Perry
Software Engineering Institute
University of Texas at Austin
perry@ece.utexas.edu

Abstract

The Invited Industry Presentations (IIP) feature leading practitioners who present and discuss problems, critical issues, and best practices of the industrial software landscape. The topics of the ICSE 2001 IIP track include empirical studies of global software development, organizational models for distributing work over many sites, challenges faced by at start-up companies, remedies for the software performance and reliability bottleneck, technology drivers for e-business, mobile phone systems and web services, methodologies for enterprise component technologies, options analysis for reengineering, and architecture-driven usability solutions.

1. Global Software Development The Bell Labs Collaboratory

Speakers: David Atkins[1], Mark Handel[2], James Herbsleb[1,] Audris Mockus[1], Dewayne Perry[3*], Graham Wills[1]

Affiliations: [1]Bell Laboratories Lucent Technologies, [2]University of Michigan, [3]University of Texas at Austin, USA

Abstract: Software development is a global enterprise for many large corporations. Searching for talent across national boundaries, and integrating groups thrown together by mergers and acquisitions are but two of the many forces conspiring to change the organizational context of software development (e.g., [1]).

For the past three years, a group of researchers from Bell Labs and the University of Michigan have been working to understand and address global development issues. The project has four concurrent threads of activity:

Empirical studies of global development. The problems of global development are varied. We conducted over 200 structured interviews at 14 sites on three continents, with people at all levels in the organization from developers to executives. In addition to the obvious problems of time zone, limited bandwidth connectivity, language and culture differences, we found the chief victim of global development to be speed [4]. Changes that cross sites take much longer than changes that are all at a single site. The difference appears not to be due to the size or complexity of cross-site changes, but rather to communication and coordination issues.

Most pressing among these communication and coordination issues [3] are 1) what we are calling issue resolution paralysis, induced by the inability to identify the right person, initiate communication, and have an effective interchange, and 2) a complete lack of informal "corridor talk" among people at different sites, which results in a surprisingly powerful impediment to the flow of information.

Collaboration tools for awareness and communication. In order to address these problems, we have developed several tools for collaborating over distance in software development. The tools include

- Experience Browser, which allows the user to explore data in the change management system through a visual interface to find people experienced if various parts of the code.
- Rear View Mirror, an instant messaging tool that supports persistent team chat rooms, a lightweight tool for informal conversation.
- CalendarBot, a web-based multi-user calendar tool that helps keep everyone informed of items such as travel and vacation plans.

All of these tools are currently in use within Lucent.

Organizational models for distributing work over sites. There are many possible ways to distribute development work over sites (see [2, 5]). For example, one might

- develop different subsystems at different sites,
- execute different process steps at different sites,
- develop a core product at a single site, and customize for different markets and customers at satellite sites,
- locate different maintenance releases at different sites.

These techniques require different mechanisms for coordinating the work, and differ in the circumstances in which they are most likely to be effective.

Best practices for global development. We have observed a number of practices that have been quite effective in improving communication. Among them are practices about

- establishing trust across sites by being even more responsive to remote colleagues than to local ones
- setting up liaisons at each site to facilitate cross-site communication
- establishing etiquette for answering e-mail and voice mail messages in a timely way
- planning for travel early in the relationship between sites – everything will work better after people have met.

Our future work will focus on continuing to introduce tools and models to practice, and to measure the results with analyses of MR data and survey results.

References

1. Carmel, E., Global Software Teams. 1999, Upper Saddle River, NJ: Prentice-Hall.
2. Grinter, R. E., J. D. Herbsleb, and D. E. Perry. The Geography of Coordination: Dealing with Distance in R&D Work. in *GROUP '99*. 1999. Phoenix, AZ
3. Herbsleb, J. D. and R. E. Grinter. Splitting the Organization and Integrating the Code: Conway's Law Revisited. in *21st ICSE 1999)* Los Angeles, CA: ACM Press 85-95.
4. Herbsleb, J. D., *et al.* An Empirical Study of Global Software Development: Distance and Speed. in *ICSE*. 2001. Toronto, Canada: IEEE Press
5. Mockus, A. and D. M. Weiss. Globalization by Chunking: A Quantitative Approach. *IEEE Software* January - March, , 2001.

* Work done while at Bell Labs.

Session Chair: Dewayne Perry, University of Texas at Austin, USA

2. Does more necessarily mean better? The Software Performance and Reliability Bottleneck

Speaker: Mantis Cheng

Affiliation: ACD Systems, Canada

Abstract: Processor speed has been increasing dramatically for the past 20 years. On the other hand, software is getting bulkier and slower. Many popular software applications have a lot of 'bells and whistles'; most users know less than 5% of all supported features. Users don't see performance improvement as processor speed increases. We are no better off than we were 15 years ago.

As a discipline, software engineering is young and full of promise. However, with current software practices, if we asked software engineers to design cars, we would have to restart our cars every mile (unreliable), fill up our tanks every hour (inefficient), and upgrade our engine control software every week (feature-laden).

What's wrong with the software industry? What have we not taught our graduates about software design? Is there a much deeper problem in our software engineering culture? In order to produce better software, does it always mean that we need to add more and more features? To release a product ahead of the competition, should the users be the beta-testers? This endless cycle of field upgrades must stop.

Let us discuss why we need a deeper appreciation of software performance and reliability. What are the bottlenecks? What are the remedies?

Session Chair: Dewayne Perry, University of Texas at Austin, USA

3. Software Engineering in a Startup

Speaker: Aleta Ricciardi

Affiliation: Valaran Corporation, USA

Abstract: Startups face enormous demands that affect the rigor with which they might adhere to software engineering and quality plans. Investors are impatient to see product, existing competitors have momentum and presence that must be countered, the processes and quality metrics are being defined along with, if not behind, product specification and development, and the development organization can often double within one or two weeks. Processes cannot assume a status quo, or even a predictably changing environment.

This talk will present lessons from the trenches: process definition, knowledge capture, product delivery, educating QA, company structure, and some interesting —possibly heretical—observations.

Session Chair: Dewayne Perry, University of Texas at Austin, USA