Conference Report: PROFES2005

Tony Dale

June 28, 2005

General

PROFES2005 is an international conference on product-focused software process improvement. There were about 120 participants. Each paper submitted was reviewed by three reviewers, with 41 out of 67 papers accepted. The conference proceedings are published by Springer [1].

PROFES is an industry/academic conference, but compared to NZ the industry people were very highly qualified; I was struck by the apparent ease with which European IT professionals can move back and forth between industry and academia.

The conference was held in the IT Dept of Oulu University, Finland. This is a sleek, modern and colour-matched facility. The conference organization was excellent, my only criticism being that I wish they had posted the "PROFES2005 Survival Guide" on the website a few weeks ago. The location and scheduling information in this little presentation would have helped me a lot on the first day of the conference—conference organisers take note!

Papers

Papers were presented in 20–minute slots. Most of the presentations garnered only one or two questions, and that was often from the session chair. Generally, there was little interaction with the audience. The quality of most presentations was good, although English as a second language hampered one or two. The most annoying aspect was that quite a few presenters really needed a laser pointer, but didn't have one. Here is a synopsis and a few opinions about the sessions I attended:

Tuesday, June 14th

Keynote: Competitive Product Engineering: 10 Powerful Principles for winning product leadership, through advanced systems engineering, compared to 10 failure paths still popular in current culture

Tom Gilb.

This keynote was entertaining and provocative rather than thought–provoking. Tom has gotten together a bunch of well–known agile Software Engineering techniques, claimed them as his own and christened them "EVO Software Engineering". While Tom is correct when he says that modern Software Engineering techniques are poorly used by industry, it doesn't follow that you could throw them all away and do no worse than at present. There were a few sweeping statements like that in this talk. Another one was: "even a bad metric is better than no metric", which might be controversial.

Many of the presenters subsequently referred to Tom in their presentations, so he made an impact.

A Qualitative Methodology for Tailoring SPE Activities in Embedded Platform Development

Enrico Johansson, Josef Nedstam, Fredrik Wartenberg, Martin Host

A worthwhile approach to tailoring software engineering processes for local requirements, using performance metrics (mostly response–time based) to optimize. Validation and evaluation of the model is done throughout the tailoring cycle.

A Rendezvous of Content Adaptable Service and Product Line Modeling

Seo Jeong Lee, Soo Dong Kim

Academically the weakest presentation I saw, this work appeared to be at an early stage of development. The aim was to create a framework for more effective, flexible and reusable software services, but how they did this wasn't well explained in either the presentation or the paper. An example would have made more sense of it—and showed the approach could work.

A Framework for Linking Projects and Project Management Methods

Tony Dale, Neville Churcher, Warwick Irwin

My reason for being there. After my presentation, one or two people sought me out to argue with me, or pick my brains.

A Meta-model for Requirements Engineering in System Family Context for Software Process Improvement using CMMI

Rodrigo Ceron, Juan C. Duenas, Enrique Serrano, Rafael Capilla

A worthwhile approach to requirements engineering, with an implementation (an Eclipse plugin). The approach boils down to a complicated meta–model for requirements documentation, relying on use–cases.

Framework for Integrating Usability Practices into the Software Process

Xavier Ferre, Natalia Juristo, Ana M. Moreno

HCI meets Software Engineering processes. The authors selected 35 HCI techniques from 94 they evaluated, for applicability to Software Engineering. There is an implementation of their approach on the web. Interviews with developers who had used their approach were supportive—the developers knew whether what they were doing had HCI implications.

Issues in Software Inspection Practices

Sami Kollanus

Software inspection (ie: desk checks) is acknowledged as important, but is often little–used or badly used. An interview–based case study of two organizations, it was found that organization size, problem size and inspection maturity affected the amount of code inspection.

Risk-based Trade-off between Verification and Validation—An Industry-motivated Study

Kennet Henningsson, Claes Wohlin

This approach used two factors: the amount of coupling between classes and the familiarity of the required functionality, to create recommendations for tailoring Verification and Validation in software engineering processes. A case study used structured interviews for evaluation, and it was found that resource utilization improves by adapting the Software Engineering process in this way.

Investigating the Impact of Active Guidance on Design Inspection

Dietmar Winkler, Stefan Biffl, Bettina Thurnher

Desk checks again. These might be done in an ad–hoc way, or using checklists which might be generic or tailored to the application, or using desk checks based on the use–cases on which the code was based. Tailored checklists and use–cases are "active guidance". A study was made of 127 Software Engineering students, and usage–based inspections were found to be the most effective (total number of defects found) and efficient (defects found per hour).

Wednesday, June 15th

Keynote: From products and solutions to end-user experiences

Ari Virtanen

A good marketing presentation (they do exist!). Apparently mobile phones are in a transition from technical, feature–based marketing to benefit–based marketing which takes in the whole product cycle, from initial purchase through service and eventual replacement. The idea is that users can be made to buy, and even pay a premium for, a technically second–rate product—think Harley–Davidson motorcycles.

Mega Software Engineering

Katsuro Inoue, Pankaj K. Garg, Hajimu Iida, Kenichi Matsumoto, Koji Torii

A new concept but not new technology. This is a low stress/low burden approach (for programmers and managers) to collecting information about a software project and, for example, applying data mining (eg: latent semantic analysis) to it. In this way, individual knowledge can be easily shared with a community of developers. See `www.empirical.jp` and `www.zeesource.net`.

Software Development and Experimentation in an Academic Environment: The Gaudi Experience

Ralph-Johan Back, Luka Milovanov, Ivan Porres

Gaudi is a student–staffed software factory which operates over the 3–month summer breaks. Students apply to work in the Gaudi factory, and the good ones are selected. The software projects may be software for researchers, or actual software experiments; the clients are departmental or University clients, but industry collaboration is a possibility. A modified Extreme Programming methodology is used. Projects are typically 1-2 person–years, or 4–6 students for 3–6 months; larger projects are broken down to this size. Apparently, having nice surroundings makes a noticeable difference to the smooth functioning of projects.

Evaluation of Three Methods to Predict Project Success: A Case Study

Claes Wohlin, Anneliese Amschler Andrews

This study examined data from the the NASA Software Engineering Lab, and used a statistical approach to examine whether three different methods could predict project success. A sliding window of projects was examined: the history of 20 projects was used to predict the next five. Of the three methods, the "critical success factors" approach showed significant ability to predict project failure—ie: there were "critical failure factors". The number of factors changed over time: there were nine, then 2, then 4.

An XP Experiment with Students Setup and Problems

Thomas Flohr, Thorsten Schneider

This was a study of groups of students which aimed to determine whether test–driven coding was better/faster than a code–then–test approach. The intention was to use an industry–like environment, but with students, who are cheaper to experiment on than professionals. Some of

the difficulties encountered were that all the students were in the same room, so could plagiarize each others work, and the difficulty of finding a good test task. The experiment concluded that test–driven coding was faster, with better coverage of the software being tested.

Views from an Organization on how Agile Development Affects its Collaboration with a Software Development Team

Harald Svensson, Martin Host

Extreme Programming has an ideal scenario, but in reality you often have such factors as a large software development group, maintenance of legacy systems, etc. This was a case study of the introduction of Extreme Programming into an organization, but while most XP studies have been of programming teams, this study attempting to get the viewpoint of the rest of the organization. The results were that while XP didn't affect organization/developer communication, the developer team gave better answers to the organization, and was perceived as being more competent, so that the organization trusted the developers more. The presentation concluded with some "agile" advice:

- The introduction of a new process like this must be supported by top management. It's also essential to address the concerns of people who oppose the changes.
- Present results of the introduction of the new process early and often. This increases curiosity about and awareness of the exercise.

Adapting PROFES for Use in an Agile Process: An Industry Experience Report

Andreas Jedlitschka, Dirk Hamann, Thomas Gohlert, Astrid Schroder

PROFES is a big software engineering method a bit like the Rational Process. This was an attempt to make PROFES more "agile". It was found that the short iterations of agile methods don't allow much time for the time–consuming PROFES process—so don't do that, then! The solution is only to use the heavyweight parts of PROFES when really necessary and the costs are justified.

Agile Hour: Teaching XP skills to Students and IT Professionals

Daniel Lubke, Kurt Schneider

What a great idea! This approach teaches Extreme Programming skills with a lecture, then a one–hour lab. Instead of teaching XP with the distraction of programming, the project is implemented with Lego—originally the project was drawing, but drawings are too loosely constrained compared with Lego. A similar approach was tried with IT Professionals, and the results were rather different than with students: the professionals tended to argue about the "contract", and pair groups tended to expand to a team/talkfest.

Panel Discussion: Incremental project management in business sense: How to manage incremental software development in dynamic a environment

In this session, four panelists, Tom Gilb (Consultant), Marek Leszak (Lucent), Otto Vitner (Delta Consultants) and Rob Kommeren (Philips) gave their takes on this problem.

Tom and Otto's talk were non-specific advice (ie: what you should do in this situation) but the other two related experiences from their companies (what they did do in this situation). In the latter cases, one or even several highly tailored project management methods had been adapted from, for example: the Spiral software development method.

By comparison, the consultants came up with rules of thumb, saying you should make each project iteration 2% of the total project timeline or budget, or that agile method-based projects

typically need the architecture rewritten after three or four iterations. The company people pointed out that they have such real-world constraints as 12-month lead times for product planning, and the consultants responded along the lines of "you have to think outside the box".

All very interesting, and a stimulating end to the conference, but we should bear in mind that the majority of software developers are at the lowest level of the CMMI model, and can't begin to deal with the process concepts addressed in this conference.

References

[1] A Dale, N Churcher, and W Irwin. A Framework for Linking Projects and Project Management Methods. In Frank Bomarius and Seija Komi-Sirvio, editors, *PROFES2005*, volume 3547/2005, pages 84–97, `http://profes2005.oulu.fi/`, June 14-15 2005. Springer.