Position Paper for OOPSLA 2006 Workshop
Escaped from the Lab: Software Practices in Large Organisations

Craig Anslow
Victoria University of Wellington, New Zealand
craig@mcs.vuw.ac.nz

1. Ideas

Before a module, subsystem, component or piece of code can be reused it must first be understood. Software visualization can help people to understand software. Software visualization is the use of the crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software [1].

In a recent survey [2] based on questionaires completed by 111 researchers from software maintenance, re-engineering and reverse engineering, 40% found software visualization absolutely necessary for their work and another 42% found it important but not critical. 7% think that is is at least relevant and 6% that they can do without but it is nice to have. Only 1% believe software visualization is not an issue at all. 4% did not answer the question.

Why has software visualization predominantly remained in the field of academic research? What are the barriers stopping software visualization techniques and tools moving from academia to software development practices in large organisations?

2. Other Questions

- Project Sales and Delivery: Should people who sell new software development projects to customers be involved in the delivery of the software? If so, will this help sales people to understand what is required to deliver a project and enable them to better forecast software projects in the future?
- Outsourcing: How does culture and spoken languages affect the software development process when requirements gathering and user interface design are outsourced to people from a different culture and speak a different language? How do these issues affect the project lifecycle?
- Methodology: When large organisations use a software development methodology such as RUP [3] or XP [4] which methodology is most applicable? Is it more applicable to pay strict adherence to a methodology or combine the best features of a methodology with an organisations own processes?
- Team Size: What is the optimum size of a software development team? How many developers, architects, testers, analysts, and project managers does it take to deliver a successful project on time and on/under budget?
- Team Formation: Large scale projects usually last many months. How does an organisation know that they have the right person for the right position in a team and does it matter? When someone leaves from a project should anyone replace that person? If an architect leaves should a developer who knows the project inside out or a new architect that does not know anything about the project replace them?
- Team Communication: How do daily meetings, email, instant messages, wikis, and informative workspaces [4] help improve team communications in the software development process? Is there a lack of or too much communication amongst team members working on projects in large organisations?

3. Experience

Craig Anslow has recently escaped from the lab and has a BSc and BSc (First Class Honours) degrees in computer science from Victoria University of Wellington, New Zealand. Craig is currently doing a MSc in computer science degree in the area of software visualization. The current title of the thesis is Evaluating X3D for use in Software Visualization.

Craig has four years commercial experience working as a web developer for Victoria University of Wellington, self employed web developer, and a software developer for a large US organisation in New Zealand.

References

[1] J. Stasko, M. Brown, and B. Price. *Software Visualization*. MIT Press, 1997.

[2] R. Koschke. Software Visualization for Reverse Engineering. In S. Diehl, editor, *Revised Lectures on Software Visualization, International Seminar*. Springer-Verlag, 2002.

[3] P. Kroll and P. Krutchen. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. Addison-Wesley, 2003.

[4] K. Beck and C Andres. *eXtreme Programming Explained: Embrace Change*. Addison-Wesley, 2004.

OOPSLA'06 October 22–26, 2006, Portland, Oregon, USA.
Workshop - Escaped from the Lab: Software Practices in Large Organisations.