ESPOO 2006 VTT SYMPOSIUM 241

Outi Salo, Pekka Abrahamsson & Päivi Järing
(eds.)

The 7th International Conference on eXtreme Programming and Agile Processes in Software Engineering

Tutorials, Workshops, Activities,
and Keynote Speeches

VTT SYMPOSIUM 241

Keywords: software engineering, eXtreme programming, agile processes, adaptive processes, agile development, software development, XP projects, testing

The 7th International Conference on eXtreme Programming and Agile Processes in Software Engineering

Tutorials, Workshops, Activities, and Keynote Speeches

Oulu, Finland, June 17–22, 2006

Edited by

Outi Salo, Pekka Abrahamsson & Päivi Jaring
VTT Technical Research Centre of Finland

Organised by

VTT Technical Research Centre of Finland
University of Oulu

ISBN 951–38–6305–0 (soft back ed.)
ISSN 0357–9387 (soft back ed.)

ISBN 951–38–6306–9 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1455–0873 (URL: http://www.vtt.fi/publications/index.jsp)

Copyright © VTT Technical Research Centre of Finland 2006

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 3, PL 1000, 02044 VTT
puh. vaihde 020 722 111, faksi 020 722 4374

VTT, Bergsmansvägen 3, PB 1000, 02044 VTT
tel. växel 020 722 111, fax 020 722 4374

VTT Technical Research Centre of Finland
Vuorimiehentie 3, P.O.Box 1000, FI-02044 VTT, Finland
phone internat. +358 20 722 111, fax + 358 20 722 4374

VTT, Kaitoväylä 1, PL 1100, 90571 OULU
puh. vaihde 020 722 111, faksi 020 722 2320

VTT, Kaitoväylä 1, PB 1100, 90571 ULEÅBORG
tel. växel 020 722 111, fax 020 722 2320

VTT Technical Research Centre of Finland, Kaitoväylä 1, P.O. Box 1100, FI-90571 OULU, Finland
phone internat. +358 20 722 111, fax +358 20 722 2320

Technical editing Anni Kääriäinen

Otamedia Oy, Espoo 2006

Preface

XP 2006 is the 7th International Conference on eXtreme Programming and Agile Processes in Software Engineering. In addition to the numerous research and experience papers as well as poster presentations, the conference comprised of a multiplicity of tutorials, activities, workshops, and keynote speeches. Together, the conference program of XP 2006 provided a unique forum for industry and academic professionals to discuss and share their needs, findings and ideas concerning Agile software development methods and solutions. This proceeding is a collection of the topics, contents, and authors of all the tutorials, activities, workshops, and keynote speeches of XP 2006.

Contents

1. Tutorials

1.1 Agile Software Development in the Large

Jutta Eckstein
IT Communication, Germany

1.1.1 Introduction

A lot of people still believe that agile software development is for small teams only. However, the agile value system and the principles behind as stated in the agile manifesto don't say anything about team or project size. Furthermore the projects I'm working on are typically large, distributed and mission-critical. Therefore, several years ago I took the challenge and tried agile software development in the large. Meanwhile I made the similar experience on many large projects: Also large and even distributed teams can benefit from a value system that is beneficial for small teams. In this tutorial I want to show how to scale agile processes to teams of 200. In fact, the same techniques are also relevant to teams of ten or more developers, especially within large organizations.

Intended audience of the tutorial are:

- Change agents and promoters of agile methods
- Executives
- Project managers, product managers, development team managers.

It would be helpful if the audience would be familiar with a specific agile process, however it's not a requirement.

Participants will recognize typical obstacles when applying agility in the large (either in a large team or/and in a large organization). They will walk away with several best practices that will help them to overcome these obstacles and with an idea how to develop their own best practices.

1.1.2 Biography

Jutta Eckstein (je@it-communication.com) is an independent consultant and trainer for over ten years. She has a unique experience in applying agile processes within medium-sized to large mission-critical projects. This is also the topic of her book Agile Software Development in the Large. Besides engineering software she has been designing and teaching OT courses in industry. Having completed a course of teacher training and led many ‘train the trainer’ programs in industry, she focuses also on techniques which help teach OT and is a main lead in the pedagogical patterns project. She has presented work in her main areas at ACCU (UK), OOPSLA (USA), OT (UK), XP (Europe) and Agile (USA).

1.2 Effective Measurement of the Software Process

Alberto Sillitti & Giancarlo Succi
Free University of Bozen, Italy

1.2.1 Introduction

Measures supply essential data in all the different engineering disciplines [1]. Such data enable and facilitate the understanding of how things work “in reality” and how to make changes to produce desired results. It has been often said that a solid and significant process improvement requires a sound measurement program [2, 3].

In software engineering, it is difficult to collect useful measures. Nearly always, metrics are collected manually with burden for software engineers (and consequently to managers) and manually collected data are often affected by errors, making them unusable [4]. The scarcity of data makes also less valuable the limited data collected, as fewer comparisons can be drawn.

Altogether, there is an apparent contradiction. On one side managers and software engineers praise measures from a theoretical standpoint, while in practice the application of measurement is limited.

Such contradiction is even more evident in the Agile Community. Agile Methods and lean management in general requires a solid measurement program [5, 6, 7]. However, collecting metrics appears to be a non directly productive activity, and, as such, it is often discarded both by managers and by software engineers.

This behavior affects the ability of researchers and software companies to perform objective evaluations of the new development methods. For instance, it is still under investigation whether and in which areas Agile Methods (AMs) perform better than traditional ones.

Measures are able to provide evidence to identify areas of applicability and a reason for managers to experiment AMs in their projects. However, it seams not possible to use the manual data collection in an effective way. Moreover this approach is against the principles of the AMs: developers have to focus only on the value for the customer, and metrics are not directly connected to it. For these reasons, a new generation of non-invasive metrics collection tools has been developed and can be used to collect data without any human effort [8, 9].

The tutorial focuses on the importance of the measures in an agile environment identifying the benefits of the approach. Product and process metrics will be considered and it will be explained how to relate the data in order to have a comprehensive view of the status of a project and identify weakness of the development process which are usually difficult to identify without such data. The tutorial will analyze traditional data collection techniques, such as the Personal Software Process (PSP), identifying their strength and weakness and how to modify them to make them compatible with the AMs. Different strategies of data collection will be presented and discussed with the participants to the tutorial in order to identify specific problems in different organizations. A tool for non-invasive metrics collection, PROM, will be presented.

This tutorial has a broad audience that includes developers and managers. Both of them are interested in the improvement of the final product and find difficult to understand if a new tool or practice is really useful or not. Collecting measures is the only way to achieve such results but traditional techniques are not designed to be applied in conjunction with AMs.

The outline of the tutorial is:

1) Goals of a measurement plan
2) Benefits for developers and managers
3) Product and process metrics
4) Traditional data collection techniques
5) Implementation of data collection in conjunction with AMs
6) Discussion of real issues with the audience
7) PROM.

1.2.2 Biography

Alberto Sillitti, Ph.D., PEng, is Assistant Professor at the Free University of Bozen, Italy. His research areas include software engineering, software metrics, component-based software engineering, integration and measures of web services, agile methods, and open source software.

Giancarlo Succi, Ph.D., PEng is Professor of Software Engineering and Director of the Center for Applied Software Engineering at the Free University of Bozen. His research areas include agile methodologies, open source development, empirical software engineering, software product lines, software reuse, software engineering over the Internet. He is author of more than 150 papers published in international conferences and journals.

1.2.3 References

[1] Fenton, N.E. & Pfleeger, S.H. 1994. Software Metrics: a Rigorous and Practical Approach. Thomson Computer Press.

[2] COCOMO, web site: http://sunset.usc.edu/research/COCOMOII/index.html.

[3] Humphrey, W. 1995. A Discipline for Software Engineering. Addison-Wesley.

[4] Johnson, P.M. & Disney A.M. 1999. A critical analysis of PSP data quality: Results from a case study. Journal of Empirical Software Engineering, December 1999.

[5] Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. & Thomas D. 2001. Manifesto for Agile Software Development. Available online at: http://www.agilemanifesto.org/.

[6] Ohno, T. 1988. Toyota Production System: Beyond Large-Scale Production. Productivity Press.

[7] Womack, J.P. & Jones, D.T. 1998. Lean Thinking: Banish Waste and Create Wealth in Your Corporation. Simon & Schuster.

[8] Sillitti, A., Janes, A., Succi, G. & Vernazza, T. 2004. Measures for Mobile Users: an Architecture. Journal of System Architectures, 50(7), July.

[9] Sillitti, A., Janes, A., Succi, G. & Vernazza, T. 2003. Collecting, Integrating and Analyzing Software Metrics and Personal Software Process Data. EUROMICRO 2003, Belek-Antalya, Turkey, 1–6 September 2003.

1.3 Test-Driven J2EE: Life Outside the Container

J. B. Rainsberger
Independent Consultant, Toronto, Canada

1.3.1 Introduction

Experience test-driven development by building J2EE components! Many students of test-driven development are introduced to the subject through simple problems, such as the classic “Money” example from Kent Beck’s Test-Driven Development: By Example. Although some people enjoy exploring new topics by tackling simple problems, there are those who would rather dive into the deep end, than merely get their feet wet. This tutorial is for those people.

The J2EE environment presents specific problems to programmers who want to practise test-driven development. When we learn to build J2EE applications, the books and tutorials we follow tend to teach us anti-testability habits. Parts of the platform themselves discourage designing for testability, and that makes test-driving J2EE applications more difficult than it needs to be. In this tutorial we'll see how to break out of that trap and build J2EE applications with all the confidence that test-driven development normally provides.

Test-Driven J2EE gives you the opportunity to practise the techniques of test-driven development with J2EE components. The goal is to build a small, but useful slice of a J2EE application while following the cardinal rule of test-driven development:

Never write a line of production code unless somewhere, a test fails.

You will add a feature to an online Coffee Shop application, giving you an opportunity to see how to test-drive new features for an existing system designed for testability.

If you have some experience with test-driven development, but have not yet satisfactorily applied the technique to the more challenging parts of a J2EE application, then you are the ideal attendee for this tutorial. We do not provide an introduction to test-driven development in this tutorial, but even if you have no experience practising it, you can certainly benefit from the demo as well as by acting as a pair-programming partner for another attendee. Also, even though this tutorial is about J2EE, past attendees have found the principles to apply equally well to the .NET platform.

1.3.2 Biography

J. B. (Joe) Rainsberger has practised test-driven development since 2000 and refined his skills by insisting on making a number of J2EE applications testable. His book JUnit Recipes: Practical Methods for Programmer Testing provides extensive examples of how to test-drive new J2EE features as well as how to test existing applications not designed with testing in mind. He has led tutorials, workshops and Open Space sessions at XP/Agile Universe 2003–2004 as well as Agile 2005.

1.4 Context Driven Agile Project Leadership

Todd Little
Landmark Graphics, U.S.A.

1.4.1 Introduction

It is common in the Agile community to talk of a development process that is “barely sufficient” to meet the needs of the project team. Based on experiences at Landmark Graphics, we observed that what is “barely sufficient” for one project may be insufficient for another, or overhead for yet another. When looking at our project history, we observed two primary attributes that influenced the type of process used: complexity and uncertainty. Complexity includes project composition such as team size and criticality, while uncertainty includes both market and technical uncertainty. Similar to the Boston Consulting Group’s Boston Matrix, we used animals to represent the four quadrants:

- Dogs – Simple projects with low uncertainty
- Colts – Simple projects with high uncertainty
- Cows – Complex projects with low uncertainty
- Bulls – Complex projects with high uncertainty.

Starting with a core set of common practices, additional practices are recommended depending on complexity and uncertainty. One benefit of this approach is identifying these project drivers and providing earlier guidance to project teams.

The tutorial develops this model and extends it to include the product lifecycle with linkages to the Boston Matrix and to the work of Geoffrey Moore.

Primary audience is designed to be Leaders, Managers, Project Managers and Executives, but all roles should be able to find value from the tutorial. People familiar with The Declaration of Interdependence for Agile Project Leadership will note two tenets pertinent to this tutorial:

- We expect uncertainty and manage for it through iterations, anticipation, and adaptation.

- We improve effectiveness and reliability through situationally specific strategies, processes and practices.

1.4.2 Biography

Todd Little is a Sr. Development Manager for Landmark Graphics. He has been involved in most aspects of software development including development, project management and general management. His focus has been on commercial software applications for oil and gas exploration and production.

He is on the Board of Directors for the AgileAlliance and the APLN and was the Program Director for ADC2004, Agile2005, and Agile2006 conference. He is a co-author of the Declaration of Interdependence for Agile Project Leadership and a founding member of the Agile Project Leadership Network. He received his M.S. in Petroleum Engineering from The University of Houston and a B.S. in Chemical Engineering from Iowa State University. He is a member of the AgileAlliance, APLN, IEEE, Society of Petroleum Engineers and is a registered Professional Engineer in the State of Texas. He has published several articles in both petroleum engineering and software engineering and has spoken at professional conferences in both disciplines. Contact him at tlittle@lgc.com.

1.5 Hands-on Teaching Agile Development

Orit Hazzan
Department of Education in Technology and Science,
Technion – Israel Institute of Technology

Yael Dubinsky
Department of Computer Science,
Technion – Israel Institute of Technology Introduction

1.5.1 Introduction

As a software team leader, how many times did you ask yourself: How shall I convey the magnificent ideas of agile development to my team? As a university professor, how many times did you try to introduce a new teaching approach to your institution? This tutorial aims at helping you achieving these goals, specifically, introducing new ideas related to software development into your organization, whether it is an academic institution or a software house. This tutorial is based on our rich experience in teaching Extreme Programming and agile practices in different settings in the industry, the army and the academia, as well as our experience in guiding their implementation in the organizations. We focus on hands-on teaching and learning agile practices to enable our participants to run these processes by themselves. The tutorial consists of three parts: the teaching of agile development in the industry, the teaching of agile development in the academia and guidance of reflective processes concern with the teaching and learning.

This half-day tutorial is intended for software practitioners from all levels. Specifically, practitioners who hold one of the following roles can benefit the most: senior software managers, project leaders, instructors of software development methods courses, instructors of capstone project based courses, facilitators of workshops. There are no required prerequisites.

The primary goal of the tutorial is to let the participants gaining hands-on experience with teaching agile practices while dealing with the conceptual and organizational changes introduced by them. This goal is achieved by an active learning approach that guides the participants to actually experience the practices as well as teaching them.

The tutorial is consists of three main parts as follows.

Part I – Teaching Agile Development in the industry
The team perspective**:** project schedule, a training program, reflective processes.
The customer perspective**:** collaboration with the customer, product quality.
Project management**:** roles, measures, case studies and analysis.

Part II – Teaching Agile Development in the academia
The course perspective**:** the role of the academia, course type, course schedule, customer, metaphors, assessment, coach training.
The student perspective**:** roles, information sharing, reflective processes.

Part III – Guidance reflective processes
Reflective processes**:** conducting reflective processes, using reflections in teaching and learning.
Bring the Agile change**:** conceptual and organizational changes, agile culture.

1.5.2 Biography

Both presenters have a significant experience with teaching Extreme Programming and agile tutorials and guiding implementation processes at the Israeli industry, army and academia. Since 2002 we present research papers and practitioner reports in all Extreme Programming and Agile conferences and in 2005 we facilitated this tutorial in the Agile Conference.

Orit Hazzan is an Associate Professor at the Department of Education in Technology and Science of the Technion – Israel Institute of Technology. In May 2004 she published her book Human Aspects of Software Engineering, co-authored with the late Jim Tomayko. She is a consultant for several software projects in the Israeli software industry. The way she combines research and practice enables her to bring a comprehensive picture into guidance processes of software teams. She presents her research in computer science and software engineering education conferences (e.g., SIGCSE) as well as in conferences that deal with agile software development (e.g., the Agile International Conference).

Yael Dubinsky is an adjunct lecturer at the Computer Science Department at the Technion – Israel Institute of Technology, and a software engineering consultant working with the Israeli army and industry. She graduated B.Sc. and M.Sc. in Computer Science from the Technion, Israel and Ph.D in Education in Technology and Science from the Technion, Israel. Her research examines implementation of agile software development methods in software teamwork, focuses on measurement of software processes and product quality. Yael has a significant experience with guiding Extreme Programming and agile implementation processes.

1.6 The Extreme TDD and Build Experience: From Acceptance Tests to Installation Kit

Erik Lundh
Compelcon AB, Sweden

1.6.1 Introduction

*[ex-per-i-ence] – NOUN ... **2a**. Active participation in events or activities, leading to the accumulation of knowledge or skill. **b** The knowledge or skill so derived. .... **5** Ward Cunningham collects experiences as part of the Eclipse Dash project.*

Q: Why should we do automated extreme TDD and Build from storytest to tested installation?

A: You will experience why and how in this tutorial!

Q: How could I get others clued on a great technology or practice?

A: By using this experience as an inspiration to develop your own experiences that bring insight and inspiration to your coworkers and organization.

Are you in an agile team or coaching a team? Does the team get the overall soft process but it is hard to motivate yourself or others in spending time and money on fully automated test-driven development flow from user stories/requirements to installation-tested ready-to-deploy product? Or do you have other areas where you look for patterns or practices that help teams and organizations "get clued" on why they should adopt a certain behaviour, infrastructure or technology?

This tutorial serves as both as an actual experience of Extreme Test-Driven Development & Build, and as an inspiration to develop your own hands-on experiences. The tutorial combines lecture with hands-on experience in Java or C# (the participant's choice). We have *experiences* prepared for C# (VS2005 with addins), Java (Eclipse, RCP, and others) and web (Fitnesse integrated with Selenium).

Available for hands-on experience: 5 Ferrari laptops with full Java and .NET development environments with popular test tools, connected to an automated build server with Subversion version control.

Bring your own laptop and connect to the build server. Open source software, limited editions and trial editions of many popular tools will be available, including Eclipse, Cruisecontrol Java/NET, Subversion, Resharper, TestDriven.NET, Fitnesse, Selenium, JUnit/NUNIT/xUnit, coverage tools, etc.

Driving development with customers tests – Another Notch on TDD

- Development driven by story tests a k a acceptance tests a k a the customers tests.
- How to automate the customers tests.
  Technical as well as "soft" issues.
- What will happen if we fail to automate the customers tests?
  Stories from the trenches.
- The TDD experience – Developing something familiar with very small steps.
- Strategies to not get bitten by refactoring and TDD when you work as a team.
- Driving development with the customers tests.
- Experience TDD driven by Fitnesse Tests.

- When does the customer tests stop being unit tests on steroids and start adding dimensions that drive architecture and generate unit test-level TDD?

Paternoster – The Continuous Build

- Jumping on the paternoster – Learn to Survive and Love the Continouous Build.
- Why is the fully automated continuous build so attractive?
- Build Server overview.
- How Cruisecontrol Works.
- Experience Cruisecontrol for .NET and Java.

Agile Version Control

- How to get the most support for agility from you CM/version control.
- The Engineering Task – A unit to commit to integration.
- Branches and Merges.

The Extreme Build

- From acceptance tests to installation kit.
- Examples of full automation with final testsuites running on an installation in a virgin virtual machine.
- Integrating virtual machines in the build.

Other tools in the build

- Web tests with Selenium.
- Style checkers.
- Coverage.

Coaches and scrum masters that want to learn more about how to develop and package *experiences* that bring insights and maturity to their teams and organizations should attend the tutorial as well as developers, coaches, scrum masters that want to improve the technical practices of their agile development:

- Agile customers that want to understand and experience how their acceptance- or story-tests can drive a flow of test-driven development with automated build and test.
- All participants will get success stories, strategies and tactics as well as hands-on experience of how it actually feels to work in the flow of a fully automated XP/Agile environment.

1.6.2 Biography

Erik Lundh has developed software for more than 25 years with experience that includes programming, design, architecture, sales, and R&D management. Erik uses XP, since 1999, as a catalyst to improve the maturity of software companies. Erik became a certified SCRUM Master in 2004, only to find that he always has done SCRUM as part of his approach to XP. Erik combines his experience as project "supertechie" with years spent advocating classic software process improvement (SPI) within the context of CMMI process improvement. Erik has experience introducing XP in organizations that range in size from small startups to large organizations. Erik evangelizes XP and Agile developments throughout Sweden. Erik is a board member of SPIN-Sweden, and an involved sponsor of most Swedish SPIN-chapters. His local chapter SPIN-SYD is the largest in Sweden, with over 40 companies including Ericsson and ABB. Erik currently lives with his wife and two kids in Helsingborg, Sweden, just across the water from the Danish castle of Kronborg, home of Shakespeare's Prince Hamlet.

1.7 Programmers are from Mars, Customers are from Venus: A Practical Guide to Working with Customers on XP Projects

Angela Martin
ThoughtWorks Limited, New Zealand

Robert Biddle
Human-Oriented Technology Laboratory, Carleton University, Canada

James Noble
Victoria University of Wellington, New Zealand

1.7.1 Introduction

This interactive and informative tutorial will introduce you to practices that will increase the effectiveness of the customer on your XP project. Customers have one of the most complex and difficult roles on a project, yet XP includes very few practices that support the customer in their role – the aim of this tutorial is to change that.

Over the last three years, we have investigated many projects around the world to identify how customers succeed in this complex and difficult task – discovering not what people think should have happened, but what really happened and what actually worked! This tutorial distils this research, grounded in practical experience, into a 3 hour session, so that by the end of this tutorial you will have gained:

- A realistic understanding of the complexity and difficulty of the XP Customer role.
- An understanding of the key roles required on a customer team, both what they are and why they matter.
- An understanding of the nine practices that enable customers to sustainably drive XP projects to successful completion – think "XP practices" BUT for customers.

This tutorial is aimed at anyone – but most specifically programmers, customers, analysts, testers and project managers – who wants to improve the effectiveness of the Customer within XP. Participants should have a background in XP (or another Agile methodology).

1.7.2 Biography

Angela Martin, ThoughtWorks Limited: Angela Martin is a consultant with eleven years of professional software development experience; she works directly with programmers and customers on agile projects to deliver software that works. She is also completing her PhD research at Victoria University of Wellington, New Zealand, supervised by James Noble and Robert Biddle. Her research utilises in-depth case studies of the XP Customer Role, on a wide range of projects world-wide. Angela is also an Agile Alliance Board Member.

Robert Biddle, Human-Oriented Technology Laboratory, Carleton University: Robert Biddle is Professor of Human-Oriented Technology at Carleton University in Ottawa, Canada. His research and teaching is in software engineering and human-computer interaction; he earlier worked as a software developer and as a technical consultant. He is widely recognised as an excellent teacher, presenter, and educator.

James Noble, Victoria University of Wellington: James Noble is Professor of Software Engineering at Victoria University of Wellington, New Zealand. He has extensive experience lecturing, teaching, and mentoring software design, software visualisation, user interface design, and design patterns, and many other topics. He has presented many tutorials at conferences including OOPSLA, JAOO, TOOLS, OzCHI, and VL/HCC.

1.8 From Concept to Cash: Deliver Fast

Mary Poppendieck

1.8.1 Introduction

When an industry imposes a compromise on its customers, the company that breaks the compromise stands to gain a significant competitive advantage. In software development, we tell our customers that high quality software takes a lot of time. There is a significant competitive advantage awaiting the companies that break this compromise. This tutorial will show how to organize software development so that value flows rapidly from concept to cash, for both new systems and on-going support. Lean Software Development moves beyond agile practices by focusing on the speed of delivery. Counterintuitive tactics such as aggressive development of multiple options, dramatic shortening of lists and queues, and responsibility-based scheduling are at the heart of going fast. Counterintuitive measurements such as cycle time replace localized productivity measurements. A counterintuitive focus on product development replaces a project-oriented approach. This tutorial will give participates a framework for organizing software development to compete on the basis of speed. It will show what works, what doesn't and why.

Consider your daily newspaper. It produces a new product every day, and nothing ever stops it. Even during the height of the damage from Hurricane Katrina, the New Orleans Times-Picayune never missed a deadline. Newspapers are put together by page editors, who always have an assortment of options for their pages as the publishing deadline approaches. No one thinks that every single story written for the paper has to get published. No one pre-plans the news pages a day or a week in advance. An option-based approach to development gives us the optimum product configuration even as it guarantees that we will have the product ready on time.

How many defects are in your defect list? How long is your list of things to develop? Has it ever occurred to you that these very lists may what's slowing down you ability to get things done? Queuing theory is easy to understand if you've ever been caught in a traffic jam: if you try to push more stuff through a pipeline that it has capacity to handle, then everything will slow down dramatically. We understand this when we have servers to manager – we seem to forget that it also applies to development work. The first law of going fast is to ruthlessly ***limit work to capacity***. The way you do this is to stop letting yourself maintain long lists of defects and queues of work to be done.

Responsibility-based planning and control means that you establish integrating events, define what is expected to be accomplished by each event, and then forget about it. Everyone knows what their job is, and everyone – always – does what is expected by the time the integrating event occurs. Impossible? First of all, you have to have an options-based approach, and then you have to limit work to capacity. Then you can start thinking about responsibility-based schedule management, where everyone is expected to pull work out of a short queue, cause dependent tasks to happen, and meet their commitments. When it works, it works much better than push scheduling.

The most effective thing you can to do to implement lean software development is to change the measurements. Most of the measurements typically used in software development are counterproductive. Measuring productivity by lines of code or function points per hour simply encourages developers to create complex code bases, when the best productivity is delivered by teams that ***write less code***. Queuing theory exposes the fact that trying to maximize "resource" utilization will inevitably *decrease* utilization.

In lean organizations, the system-level measurements that drive the right behavior at the sub-system level are: Cycle Time, Financial Return, and Customer Loyalty.

This tutorial will be of interest to managers and technical leads that are familiar with the basics of agile software development.

1.8.2 Biography

Mary Poppendieck has been in the Information Technology industry for thirty years. She has managed solutions for companies in several disciplines, including supply chain management, manufacturing systems, and digital media. As a seasoned leader in both operations and new product development, she provides a business perspective to software development problems.

A popular writer and speaker, Mary's classes on managing software development have been popular with both large and small companies. She is co-author of the book Lean Software Development: An Agile Toolkit, published by Addison Wesley in May, 2003 and winner of the Software Development Productivity Award in 2004.

1.9 Collaborative Workplaces: Creating an Open Environment for Agile/Adaptive Projects

Pollyanna Pixton
Evolutionary Systems, Salt Lake City, Utah, U.S.A.

Diana Larsen
FutureWorks Consulting, LLC, Oregon, U.S.A.

1.9.1 Introduction

Agile flourishes in collaborative workplaces with open environments. Self-organizing teams thrive in organizational climates that stimulate communication, foster working relationships, encourage feedback and strengthen innovative thinking. Creating these open environments requires attention to a collaborative model as well as continuing focus on small, daily interactions and information flows. Business prospers when teams can collaborate and contribute.

In "Collaborative Workplaces" Pollyanna Pixton and Diana Larsen define the key factors involved in creating an open environment through developing and sustaining collaborative workplaces. The workshop includes opportunities for practice in the collaborative process and in other techniques to unleash the talent on teams. The presenters demonstrate ways to foster open dialogue that stimulates innovative development and improved processes, operations, products and services. Participants will develop specific strategies for creating collaborative workplaces in their work situations, including an action plan to begin the transition.

This tutorial will combine short interactive exercises with lecture and discussion. We will intersperse the introduction of key content pieces through lecture/description with experiential activities to reinforce the concepts. We also rely on audience participation in sharing knowledge and create a collaborative learning environment.

This tutorial is targeted to audience of: CxO's, Vice Presidents of Development, Product or Program Managers, Project Leaders and Team Leaders with an interest, or some experience, in leading in Agile environments.

1.9.2 Biography

Pollyanna Pixton is widely recognized for her ability to lead the collaborative efforts of talented people who want to expand what they already do very well to being even better. She works with executives and teams inside corporations and organizations to improve their productivity and effectiveness to achieve lasting results using collaboration. She founded Evolutionary Systems in 1996, and brings over 35 years of executive and managerial experience from a variety of successful business and information technology ventures to her work as a consultant. Ms Pixton's education includes a Master's degree in Computer Science, three years of graduate studies in Theoretical Physics, and a Bachelor's degree in Mathematics.

She was primarily responsible for managing the building of the Swiss Electronic Stock Exchange, developing sophisticated control systems for electrical power plants throughout the world, and merging the complex technologies and data systems of large financial institutions. Her background includes leading the development of complex e-commerce projects, real-time applications, positioning systems, and generating original computational research.

Pixton co-founded the Agile Project Leadership Network (APLN) and serves on that Executive Board. She chairs the APLN Leadership Summit for 2006 and in Agile 2005 she presented a tutorial and workshop on becoming and agile leader. In 2004, she chaired the ADC Executive Summit and held the Getting Leaders Onboard workshop at XP Universe. www.evolutionarysystems.net.

Diana Larsen concentrates on building people's capability to interact, self-organize and shape an environment for productive teams. Through her work, Diana demonstrates a unique blend of expertise, proficiency, candidness and compassion. Past clients recognized her as the consultant who set a standard for future working relationships. Her employees named her "the best boss ever", and colleagues identify Diana as an exceptional facilitator.

In her consulting practice, Diana unites with leaders and teams on software development projects to strengthen their ability to support and sustain change initiatives, improve project performance, and retain organizational learning. She collaborates with her extended network of extraordinary colleagues to bring the best combination of talents to every project. Diana co-presents the workshop, "Secrets of Agile Teamwork: Beyond Technical Skills" and co-authored the book on retrospectives for Agile development with Esther Derby. She presents workshops on collaborative executive and project leadership with Pollyanna Pixton. Diana Larsen speaks and writes on subjects including team deveopment, project leadership, Agile methods, retrospectives, and organizational change. http://www.futureworksconsulting.com.

1.10 Agile System Testing with Texttest and xUseCase

Geoff Bache
Carmen Systems, Gothenburg, Sweden

Emily Bache
AstraZeneca, Mölndal, Sweden

1.10.1 Introduction

Tests in XP have traditionally been divided into "Unit tests" and "Acceptance Tests". However, it is also possible to think about them having two independent dimensions: their scope (unit or system) and their primary owner (developer or customer).

xUnit has long been popular in the developer/unit quadrant and many have extended it to higher granularity for system integration testing. The success of Fit for testing business rules has shown that it is possible to create customer-relevant tests at a level well below the whole system. But there is still something of a lack of a widely accepted agile tool for customer testing at the system level. This tutorial aims to help fill that gap.

We also offer the approach as an excellent solution to testing some types of system where other existing tools either break down or have severe drawbacks. These include:

- Legacy Systems
- Scripts and other non-interactive systems (particularly those that create, edit and delete files)
- CPU-intensive systems.

The Approach and Philosophy

Perhaps the key insight offered is that in the agile world the ‘correct behaviour’ of a whole system can change a great deal over time. It follows that agile system testing should primarily be about ‘behaviour change management’ rather than ‘correctness assertion’. The aim of the approach is therefore to create a representation of the system behaviour independent of the system code and source-control this alongside the code.

Therefore, both user choices and system responses are represented as domain-language plain text files, detailed enough to ensure that system behaviour does not change unintentionally. When behaviour changes intentionally, developers update these files and indicate the reason for the change. A complete traceable history of system behaviour and the reasons for the changes to it is thus created as a by-product.

The Tools

As for the tools, TextTest is a testing tool that tests an application by running it from the command line and filtering and comparing textual output that it produces. It is expected that the developers will use a logging framework (such as the log4x family) to write informative high-level textual decriptions of its behaviour that can be easily understood by non-technical people. Changes in this output are then flagged as test failures, but can be easily saved and checked in when behaviour is intentionally changed.

For GUI applications, the command line is not enough: you need to be able to simulate the actions of system users in some way. This is where xUseCase comes in. It is a family of similar tools for simulating the actions of a user on a GUI, record/playback style. It is different from most record/playback tools in that it records a high level “use case” in the language of the domain. This is achieved by developers assigning a domain language name to each type of action that can be performed with the GUI and inserting this mapping into xUseCase.

More information on the tools can be found at http://www.texttest.org.

On Legacy Systems

The approach eliminates the need for the system under test to present an API for tests to call into, which many other agile testing tools require. Such APIs are a major headache for testing legacy systems because they require you to make substantial, risky changes to the system under test before tests can be introduced. This relationship between tests and system under test is not eliminated, but reversed: required code changes call exclusively out from the system under test and are stubs only in a non-test environment. These are simple and very low-risk to introduce.

On File-Changing Scripts and CPU-Intensive Systems

Many test tools assume that interaction modelling of some sort will be needed, either driving a GUI or calling an API. Where this is not present, TextTest can easily and naturally be used alone.

Because plain-text files produced by the system under test are what TextTest handles anyway, it has mechanisms for ensuring that all file changes occur in an isolated environment, eliminating any need to do this work in the script under test, and simple ways to collate and compare the produced files.

Where tests are CPU-intensive running them all in series on a single machine is generally not an option. TextTest integrates smoothely with the grid engine products “Sun Grid Engine” and “LSF” which distribute the tests over a network so that they run in parallel. Even if individual tests are not CPU-intensive they can be as a group if there are lots of them, and this ability radically increases the frequency with which they can be run.

Software developers, architects and testers will all benefit from learning about this innovative testing approach, both in theory and in practice.

1.10.2 Biography

Geoff Bache is an experienced software developer and XP coach, working for the software product company Carmen Systems in Gothenburg, Sweden. He has been interested in XP Acceptance Testing since 2000 and working on developing the approach that has become TextTest and xUseCase since then. He has presented papers on Acceptance Testing at the XP2003 and XP2004 conferences. At XP2005, he co-chaired two successful workshops together with Rick Mugridge (of the Fit tool) and Brian Swan (of the Exactor tool). He maintains the site www.texttest.org.

Emily Bache currently works for AstraZeneca in Mölndal, Sweden, writing applications in python to help scientists with their research. She has been practicing Test Driven Development with xUnit since 2000, and with TextTest since 2002. Emily has presented posters about automated testing and writing software in collaboration with scientists respectively at two previous XP conferences. More recently Emily has started a regular coders dojo at her workplace, with the aim of teaching and learning Test Driven Development.

1.11 Expressing Business Rules as Executable Specifications

Rick Mugridge
Rimu Research Ltd, New Zealand

1.11.1 Introduction

Learn how to express business rules as storytests [1], with a focus on expressing the business domain with clarity and brevity. Storytests serve to clarify and communicate the domain as *executable specifications* and to aid in developing the system and testing it in a variety of ways.

Writing storytests is usually complicated by several factors: The business domain needs to be understood, and often needs to be clarified [2]. Storytests need to evolve to help this understanding evolve. Emphasis is often placed too early on the detailed testing aspects, rather than on expressing the business domain as clearly as possible. The storytests often make premature commitments to details of the application being developed, or are not written until those details are known.

This tutorial will give you experience in expressing business rules well as storytests with *FitLibrary* [3, 4], through small group exercises. We'll see that such storytests evolve as the whole team's understanding of the business needs and the system evolve.

The audience of this tutorial are customers, business analysts, product managers, project managers, testers, and programmers. Tutorial is suitable for those with no experience in storytests, right through to those experienced with Fit [5, 6] and/or FitLibrary who want to refine and advance their techniques and extend their knowledge. It is as relevant to programmers as it is to others in a project team.

After attending this tutorial, you will see how to use (or better use) storytests as *executable specifications* that are a powerful, alternative approach to agile requirements. You will have gained some experience in writing and evolving storytests, which you'll be able to apply to your own work. You will have begun to use the latest patterns, approaches and developments in storytest driven development.

The content of the tutorial is as follows: Why the business domain is important and how it is expressed directly in storytests. How concrete examples aid in understanding, clarifying, evolving, and communicating about the domain and the system. Expressing *calculation*, *constraint*, and *workflow* business rules through examples with DomainFixture tables in FitLibrary Evolving a *ubiquitous language*. Domain Driven Design patterns for storytests. How storytests serve as executable specifications, for driving development and for various types of testing.

The process of the tutorial is as follows: After introducing the main ideas, the participants will work in small groups with large sheets of paper on a series of exercises (computers won't be used). These exercises involve examples and focus on specific practices.

The outline of the tutorial is as follows:

1) Introduction to Tutorial (10 mins)
2) The main ideas. (75 mins)
3) Exercises, with guidance. Small groups will tackle focussed exercises in expressing business rules. Exercises will range in size in a variety of domains. (75 mins)
4) Retrospective on experience with exercises. Abstracting and generalising from those exercises. Open questions. Conclusions. (20 mins)

1.11.2 Biography

Rick Mugridge is the lead author of the first book on storytests: *Fit for Developing Software*, Prentice-Hall, June 2005. He has developed and is evolving *FitLibrary* to better support storytest driven development. He is a leading thinker and inventor in this area, and consults and coaches internationally in storytests, domain driven design, and agile software development. He ran tutorials at XP2005 and Agile2005. Rick runs his consulting company, www.rimuresearch.com, out of New Zealand.

1.11.3 References

[1] *Storytest* and *Storytest Driven Development* were coined by Joshua Kerievsky, http://www.industrialxp.org.

[2] Eric Evans, 2004. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison Wesley.

[3] Rick Mugridge and Ward Cunningham, 2005. *Fit for Developing Software: Framework for Integrated Tests*. Prentice-Hall.

[4] *FitLibrary*, https://sourceforge.net/projects/fitlibrary.

[5] *Fit*, http://fit.c2.com.

[6] *FitNesse*, http://www.fitnesse.org.

1.12 Planning Effectively with Stories: Requirements & Planning Made Easy

Mike Hill
Industrial Logic, Inc.

1.12.1 Introduction

If you think you already know everything you need to know about user stories, then you're ready for a pop quiz: which of the statements below is true?

1. The key benefit of user stories is that they maximize developer efficiency.
2. Working by stories is like working from a requirements document.
3. You should use a generalized user-interface scripting tool for storytesting.
4. Writing storytests is strictly a customer responsibility.
5. Stories are best estimated in units of time.

If you answered ‘true’ to any of these five statements, you should consider taking this tutorial, as you’re still working with some user story myths. Stories are a simplifying token in a complex protocol requiring domain knowledge, technical skill, and delicate human judgment. The tutorial emphasizes real-world experiences and gives the students a firm grounding in these areas:

- Why using stories is preferable to other planning techniques.
- How to write and recognize a ‘perfect’ user story.
- How to run stories through the planning process.
- How (and when) to break larger features into stories.
- How to write and implement good storytests.
- How to sort user stories to bring maximum value to their enterprise.

Because planning and estimating involves both Customer and Developer roles, this tutorial is equally suited to either side of an agile team. No previous knowledge of agile practices is required. The final section is dedicated entirely to real-world story-writing problems supplied by the audience. Students may wish to prepare in advance by thinking of the kinds of problems they anticipate in using stories inside their teams. Students will get real hands-on work with stories, confronting the same problems they’ll see when they return to work from the conference.

1.12.2 Biography

Mike Hill has been helping companies transition to XP since the beginning of the movement. His work has taken him to groups large and small all over North America and Europe. Currently, he is a senior coach working for Industrial Logic, Inc., the world’s leading experts on adopting Agile Practices. He is widely known for his direct and humorous presentation style, and for his willingness to get down in the trenches with his clients.

1.13 Agile Estimating and Planning

Mike Cohn
Mountain Goat Software, Colorado, U.S.A.

1.13.1 Introduction

There is a common misperception that agile teams do not plan. The truth is that agile teams do plan and often spend as much or more time planning than do traditional project teams. This misperception arises because agile projects do not begin with lengthy planning phases. Instead, agile projects are successively refined through both release and iteration planning as the project progresses.

It is a well established but often ignored best practice to estimate the size of a project before estimating how long it will take. Traditionally managed projects have used function points and non-comment lines of code as measures of project size. Agile software development teams estimate the size of their projects in either story points or ideal days.

Story points are a technique for estimating the combined size and complexity of a user story, feature, or other unit of work. When we estimate with story points, we assign a point value to each item. The raw value we assign to a particular item is unimportant; what matters is its value relative to other items. A story that's been assigned a 2 should be twice as big and complex as a story assigned a 1. It should also be half the size of a story assigned a 4.

An alternative unit for estimating size favored by some agile teams is the ideal day. An ideal day is a day when you work only on the task you have chosen, when you have all the necessary resources ready and waiting for you, and when you work without interruption – that is, no e-mail, no phone calls, and so on to disrupt the workflow. Ideal days happen rarely, if ever, but are nonetheless a useful unit for estimating the size of work required to develop a new feature.

Regardless of which unit of measure a team uses to estimate, a couple of guidelines apply:

1. Estimates of relative size are more accurate than estimates of actual size. That is, we are better at saying, “This is a little bigger than that and much smaller than the other one” than we are at saying, “This is 100 feet high, and doing that will take 13 hours.”
2. We are better at estimating items that are within approximately one order of magnitude.

Together, these two guidelines suggest that we use an estimation approach that encourages estimation by analogy (“This seems roughly the same size as the one we did last month”) and that the largest of the user stories or features should take no more than approximately ten times the amount of time required to develop the shortest user stories.

A very successful approach to this task is called planning poker, which is loosely based on but is an agile variant of the wideband Delphi approach. Planning poker is played at an estimating meeting and features the active participation of all members of the development team.

Equipped with a set of estimated features, an agile team prepares a release plan. A release plan indicates either approximately which features can be delivered by a fixed date or by approximately what date a fixed set of features can be delivered. To create a release plan the estimated features are prioritized by someone representing the business or users of the system under development. Given a prioritized set of features, a release plan is prepared based on expectations about the quantity of work the team is expected to complete per iteration, which is known as the team’s velocity. For example, if a prioritized set of features is estimated at a total of 240 story points and the team’s velocity is known to be 40 story points per iteration, a simple estimate of the overall duration of the project is 240/40=6 iterations.

A release plan is an excellent high-level guide. The team gets general guidance about its schedule from the release plan; but as each iteration begins, the team uses the high-level release plan to create a more detailed plan for the coming iteration. There are two primary reasons agile teams do iteration planning. First, to create a more detailed, actionable plan for the iteration. Second, to adjust the plan based on new information and knowledge.

In this way, agile planning becomes a two-stage process. The first stage is the release plan, with its rough edges and general uncertainties. The second is the iteration plan. An iteration plan continues to be rough and uncertain; but because the iteration plan is created concurrently with the beginning of a new iteration, it is more detailed than a release plan.

During iteration planning, an agile team looks into the future of the project no further than the one- to four-week iteration they are about to begin. With this short-term focus, the team can turn its attention from the features or user stories of release planning to the specific engineering tasks needed to develop a story. During iteration planning, the team thinks about each feature or story that will be developed during the iteration. For each, the team creates a list of tasks that must be performed.

There are many reasons why an agile approach to estimating and planning results in reliable plans. First, replanning occurs frequently. Agile teams do not do all of their planning upfront; rather planning is spread throughout the project. Second, agile teams acknowledge the need to plan at different levels, creating both near-term iteration plans with moderate amounts of details and longer-term release plans with less detail. Third, agile teams create feature-based rather than task-based plans. This leads to a greater shared understanding of the system to be built. Further, rather than relying on upfront, predictive resolution of scheduling and coordinating details these are left to the team to work through during the iteration. Finally, an agile approach to planning acknowledges uncertainty both about exactly what will be built and how the team will build it. Reduction of and the progressive elimination of this uncertainty becomes a key consideration in how agile teams plan their work.

1.13.2 Biography

Mike Cohn is the founder of Mountain Goat Software, a process and project management consultancy and training firm. Mike specializes in helping companies adopt and improve their use of agile processes and techniques in order to build extremely high performance development organizations. He is the author of Agile Estimating and Planning and User Stories Applied for Agile Software Development, as well as books on Java and C++ programming. With more than 20 years of experience, Mike has previously been a technology executive in companies of various sizes, from startup to Fortune 40. He has also written articles for Better Software, IEEE Computer, Software Test and Quality Engineering, Agile Times, Cutter IT Journal, and the C++ Users' Journal. Mike is a frequent speaker at industry conferences, is a founding member of the Agile Alliance, and serves on its board of directors. He is a Certified ScrumMaster Trainer and a member of the IEEE Computer Society and the ACM. He can be reached at mike@mountaingoatsoftware.com.

1.14 Team Health Indicators and Holistic Coaching

David Hussman
SGF Software, U.S.A.

Michael Feathers
Object Mentor, Inc., U.S.A.

1.14.1 Introduction

The philosophy of Agile Development is to place people before process. While this can be liberating for many teams, allowing them to meet their goals with increased responsiveness, it also makes development sensitively dependent on team health and dynamics.

In this tutorial, we will describe a series of observations about healthy teams, and key indicators that you can use to determine whether a team is living up to its potential. We will also describe techniques for addressing common team issues and ways to promote healthy agile culture within your team.

The format will be a combination of lecture (presentations) and group exercises.

Anyone working on, forming or joining an agile project will benefit from this tutorial. Healthy teams most often have more that one person fostering collaboration; if you are one these people, or would like to be, this tutorial will provide you with examples, tools, and experience.

1.14.2 Biography

Both presenters have 15+ years of software development experience, and both have chosen to use their experiences to help transition others to using agile practices and principles to build better software in ways to draw out the creative and collaborative in people.

David Hussman has presented at XP / Agile conferences for the past 5 years. He is also a full time coach and agile trainer, working in the US, Canada, Ukraine, and Russia. For the past 5 years, he has coached and training many companies of various sizes across many industries. In a previous life, David was a music producer and he draws on that experience in helping communities transition to using Agile. David is also a Cutter Consortium Senior Consultant and presenter at the NoFluffJustStuff conferences.

Michael Feathers has been involved in the XP/Agile community since its inception. While designing biomedical instrumentation software in the late 1990s, he met several of the members of the Chrysler C3 team at a conference and was persuaded by them to try XP practices. Subsequently, he joined Object Mentor where he has spent most of his time transitioning teams to XP. Michael is also the author of 'Working Effectively with Legacy Code'.

1.15 Testing in a Quasi-Agile Software Development Environment: Practical Strategies for Mixed Culture Projects

Timothy D. Korson

1.15.1 Introduction

This tutorial focuses on practical issues faced by increasing numbers of testers today. These issues arise from the fact that most test organizations are still structured around traditional software development practices even though many software development teams are heading full steam into modern agile software development techniques. QA managers trying to encourage best practices as recommended by CMMI and SPICE find themselves at odds with developers trying to adopt best practices as recommended by the Agile Manifesto. This leaves corporate QA stuck coping with an organizational and technical paradigm shift that traditional QA policies and practices are inadequate to handle.

In the highly iterative environment characteristic of these agile development projects, development and testing processes are much more tightly integrated. System testers are expected to test early immature increments of the software, and are often called upon to plan, support and review the unit and component-level testing process. Developers, in addition to unit testing, may be called upon to assist with the automation of certain system-level tests. Risk assessment and overall test asset allocation must also be adapted. In addition to the discussion of specific techniques and best practices, this tutorial addresses how to adapt, survive, and hopefully even thrive in mixed culture environments, where the developers are coming from an agile mindset, but some or all of the stakeholders, managers, testers, and others in the organization are coming from a traditional mindset.

The tutorial is targeted to:

- Testers
- Test Managers
- Project Managers
- Process improvement staff
- Business Analysts
- Clients and other stakeholders
- Developers.

1.15.2 Biography

Timothy Korson has had over two decades of substantial experience working on a large variety of systems developed using modern software engineering techniques. This experience includes distributed, real time, embedded systems as well as business information systems in an n-tier, client-server environment. Dr. Korson's typical involvement on a project is as a senior management consultant with additional technical responsibilities to ensure high quality, robust test and quality assurance processes and practices. Dr. Korson has authored numerous articles, and co-authored a book on Object Technology Centers. He has given frequent invited lectures at major international conferences and has contributed to the discipline through original research. Dr. Korson has recently presented this tutorial at a number of conferences in the USA and in Europe. This tutorial has received outstanding feedback from attendees.

1.16 Merciless Refactoring with Eclipse

Martin Lippert & Matthias Lübken
it-agile GmbH, Hamburg, Germany

1.16.1 Introduction

Since Martin Fowler has published his ground-breaking book on refactoring many things have changed, especially the day-to-day refactoring work has changed due to the comfortable IDE support. The goal of the tutorial is to provide an insight how the refactoring support that is built into common modern IDEs can be utilized to do merciless refactoring on a minute-to-minute basis. The tutorial demonstrates the refactoring capabilities of modern IDEs and lets the attendees feel how those refactoring tools can change their way to develop software. They learn how to utilize modern refactoring support to do improve the design of their software systems all the time.

Aside of the very basic and small refactorings we will explain how more complicated refactorings can be realized using the IDEs refactoring support. E.g. we show how to use the often overlooked "Change Method Signature" refactoring to do even complex refactorings in a few minutes, we will demonstrate how to use "Inline Method" to remove deprecated method calls and "Inline Constructor" as a combination of "Introduce Factory" and "Inline Method".

Special attention is paid to the boundaries of modern refactoring support within integrated development environments. Especially those refactorings that are supported in a non-safe way will be uncovered. This will protect attendees from blindly using these automated refactorings while changing the behavior of the system at the same time.

The tutorial is targeted to practitioners: developers and architects. The attendees will get an insight how merciless refactoring can be realized using the refactoring support that is built into the Eclipse IDE. Best practices ranging from nice keystrokes to tips and tricks to combine small refactoring features will be demonstrated in a way to be used right away. We will refactor a system with code smells in front of the audience. The tutorial will contain pair programming by the two instructors and pair programming together with the audience. While we refactor the system we explain what we are doing and introduce the IDE refactoring features. Attendees should have a solid understanding of object-orientation. Basic understanding of refactoring or Eclipse is not required.

1.16.2 Biography

Martin Lippert is a consultant and coach for software engineering technology and agile development methods. He focuses on software development on top of the Eclipse platform, refactoring, extreme programming and aspect-oriented programming. He worked for the AspectJ project at PARC as an intern during the summer of ’99 and has given a number of talks, tutorials and demonstrations on various topics of software engineering at international conferences. He is co-author of the upcoming book on complex refactorings in large software projects that will be published by Wiley in March 2006.

Matthias Lübken is software engineer at it-agile in Hamburg. He has several years of experience with agile software projects, mainly eXtreme Programming. He focuses on development on top of the Eclipse rich-client platform and is co-author of the German book “Eclipse – The Platform”.

1.17 Information Radiation in Practice: Communication Tools for Colocated Teams

Ilja Preuß
disy Informationssysteme GmbH, Germany

1.17.1 Introduction

Communication is a cornerstone of Agile software development. Well working communication allows a team to self-organize, to keep focused, to act responsibly and to build trustful relationships, leading to more effective collaboration. With their emphasis on informal ad hoc communication, Agile teams can benefit from tools that guide communication without restricting it. Information Radiators, which make key information easily and publicly accessible, are such a set of tools.

The goal of this tutorial is to provide you with the thought models and tools to make effective use of Information Radiators. We will explore the communication needs of Agile teams and how Information Radiators can help to meet them. We will take a look at examples from real projects and discuss the qualities that made them help the team, or how they failed to do so. We will learn about different formats that are in use in the community, and how to adapt them to individual situations. One focus will be on important social and economical implications of introducing Information Radiators at the workplace. A group activity will allow us to relate the discussion to individual experience.

This tutorial will be of equal interest to developers, managers, coaches and anyone else who is part of a development team. The only prerequisite is a basic understanding of Agile software development approaches.

1.17.2 Biography

Ilja Preuß is a developer and mentor at disy Informationssysteme GmbH, a German software and consultancy service provider. His focus is on improving all aspects of the team's development process and promoting technical excellence, for a big part by driving forward the adoption of Agile practices. He is a co-author of the German book “Softwaretests mit JUnit” (Link et al. 2005) and holds presentations and tutorials at conferences and local user groups. He also is a Sheriff at JavaRanch.com, the biggest non-commercial website for Java.

2. Workshops

2.1 AOSTA – Agile Open Source Tools Academy

Werner Wild
Evolution, Innsbruck, Austria

Barbara Weber
University of Innsbruck, Austria

Hubert Baumeister
Technical University of Denmark, Lyngby, Denmark

2.1.1 Introduction

Our workshop provides a platform to share experiences, exchange success stories and discuss potential pitfalls when using Open Source Tools (OST) for Agile Development. The goal of this workshop is to create awareness of useful OST and help to improve one's portfolio of tools for Agile Development.

Everyone who already uses or plans to use OST for developing software the Agile Way can participate. All, from hard core developers via project managers to CIOs are welcome to share their experiences and expectations. In addition, OST developers should attend to gain additional insights in their "customer's" agile needs and want to better steer their ongoing open source projects. Finally, whoever wants to get a quick overview on the state of the art in OST for Agile Development can participate in the discussions and/or demos; however, we kindly ask participants to get ready to demo, or at least share some stories about their favourite tool(s). Bring your Laptop!

To get started a comprehensive overview on OST is given by the organizers when presenting the results of two Master Thesis (e.g. Value Benefit Analysis) at the MCI (Management Center Innsbruck). After this brief introduction workshop participants should present a short summary of their agile open source toolbox, including the pros and cons they find noteworthy. Then, like in an Open Space, "workshoppers" should demo their agile toolkit, or, at least, their favourite OST at given timeslots. Short “hands on” sessions would be great, if there are the “right” number of participants at each spot in the Open Space. Finally, a wrap up session with all participants will give a chance to discuss open questions, share “war stories” and get feedback.

This workshop provides participants with the unique opportunity to profit from the experience of real practitioners using OST in their current projects and, equally important, to leave with the gratifying feeling of having been able to help others with your expertise. Participants will gain a quick perspective whether a specific tool can ease their daily work and learn how to avoid well know and not-so-well-known pitfalls.

A comprehensive list of OST for Agile Warriors will be created as one of the publicly available outputs from this workshop, it will be made available on the web. However, physically present, active participants will learn the most, e.g., through the shared stories and networking opportunities. And, last but not least, you will be able to spend a fun and nice morning with great people like you!

2.1.2 Biography

Werner Wild has been in IT for almost 30 years and currently is a consultant with Evolution, Innsbruck. He also lectures at the University of Innsbruck, the University of Bolzano and the Management Center Innsbruck (MCI). He has long term experience with many practices of XP as a developer, project manager and consultant and tries hard to convince his students to become more agile!

Barbara Weber is a full-time researcher at the Computer Science Department, University of Innsbruck, Austria and specializes in Business Process Management/Business Agility. She has given lectures in Agile Methods for several years and managed numerous XP projects with graduate students. Her development projects are almost exclusively done with Open Source tools.

Hubert Baumeister is associate professor at the Technical University of Denmark, Lyngby, and is one of the few people who has been attending all (!) XP 2000–2005 conferences! In addition, he served as Program Chair of XP 2005 and as Academic Chair of XP 2004.

2.2 Introducing Agile Concepts in "Traditional" Environments

Scott Duncan
The Westfall Team, U.S.A.

2.2.1 Introduction

This half-day workshop is intended to address how to present agile concepts to people unfamiliar with any specific agile method, but who follow some process, probably based on waterfall concepts (possibly with iteration). The goal of the workshop is to develop "recommendations" for how to present agile concepts in ways that allow people in an organization to adopt them, perhaps bit-by-bit, without feeling, or making the organization feel, like some big change has to take place or some specific method has to be committed to right away. At the very least, participants will share their experiences introducing agile concepts in ways that allow them to use one another's ideas back in their own environments.

It will be desirable that attendees have some experience (not necessarily successful) trying to introduce agile concepts into their (or someone else's) environment. They do not have to have tried to bring in a full-blown, named method. The key is how they explained and tried to adopt/adapt agile principles and values to an environment employing other methods/approaches. Attendees, whether they have such experience or not, are encouraged to bring a written "position" paper with them (200–250 words) which will be collected as part of the workshop materials. Possible topic areas for such papers would include (but not be limited to) the following:

- Change Management
- Communication
- Contracts
- Delivery (frequent iterations)
- Design (refactoring)
- Documentation
- Legacy Systems (enhancement and testing)
- Planning

- Process and Assessment
- Requirements
- Roles
- Scaling Up
- Verification & Validation.

After the conference the results will be written and “published” in some way, including seeing how the material might apply to the IEEE standard. Some material may also be included as a part of a companion book on the IEEE standard for the IEEE Computer Society Press. (Such a book can have a lot of ideas, advice, experiences, etc. in it not appropriate to a more terse standard but useful for those looking to implement the standard in an effective manner.)

2.2.2 Biography

Scott Duncan has been involved in commercial and government software development since 1972. For the last 12 years he has been an internal/external consultant and trainer in software quality and process engineering. He is a member of the IEEE-CS, ACM, and ASQ. He is Standards and Web Chair for ASQ’s Software Division, and the Division’s representative to the U.S. Technical Advisory Group for ISO/IEC JTC1/SC7 and to the Executive Committee and Management Board of IEEE’s Software and Systems Engineering Standards Committee. He is also Working Group Chair of IEEE 1648 (agile methods) and IEEE 90003 (IEEE adoption of ISO 90003).

2.3 Agile Development with Domain Specific Languages

Steven Kelly
MetaCase Consulting, Finland

Alan Cameron Wills
Microsoft, United Kingdom

2.3.1 Introduction

This workshop will continue the success of last year in investigating the application of Domain Specific Languages within Agile development. A Domain Specific Language (DSL) is designed to express the requirements and solutions of a particular business or architectural domain. SQL, GUI designers, workflow languages and regular expressions are familiar examples. In recent years, Domain-Specific Modeling has yielded spectacular productivity improvements in domains such as telephony and embedded systems. By creating graphical or textual languages specific to the needs of an individual project or product line within one company, DSM offers maximum agility. With current tools, creating a language and related tool support is fast enough to make DSM a realistic possibility for projects of all sizes. And to refactor, you just change the generator!

The intended audience consists of developers and technical managers interested in finding out more about DSM. Experience of product lines, building product frameworks or creating DSLs is a bonus, but by no means necessary. Benefits of participation include a better understanding of:

- When to use DSM, and what for
- How to adjust the development process to use DSM
- How to create DSLs and use DSM.

2.3.2 Biography

Steven Kelly is CTO at MetaCase, and has been the lead on the MetaEdit+ DSM tool since 1996. He is also co-founder of the DSM Forum, has served on the committee of the OOPSLA workshops on DSM since 2001, and has been giving metamodeling and DSM tutorials around the world since 1993.

Alan Cameron Wills (Microsoft) works on DSL tools for the Visual Studio IDE. Before joining Microsoft, he was a consultant in development methods, and co-developed the Catalysis approach to software development. He has run successful workshops and tutorials in related topics at SPA2005 and OT97-04.

2.4 Patterns of Adopting Agile Development Practices

Amr Elssamadisy
Valtech Technologies

Ahmed Elshamy
ThoughtWorks, Inc.

2.4.1 Introduction

How does one go about adopting an agile practice(s)? This is a question that can be answered very well by a pattern which aggregates experiences of experts who have successfully done so in the past.

The agile development community is no longer in its infancy, this conference is the 7th XP conference and many agile methodologies have roots that go back to the mid 90's. Many of us have successfully adopted, adapted, and even dropped many agile practices on multiple projects. Many (dare I say most?) of us are no longer using 'canonical' sets of practices from given methodologies. Case studies and literature reviews have been the traditional methods of reporting this information; they are very valuable but more needs to be said about our collective experiences. Patterns are an excellent vehicle for aggregating experiences; and we currently have the experiences necessary to write real patterns communicating our experiences in adopting agile practices. The work in this workshop will be seeded with patterns developed at ChiliPlop 06 earlier this year and our results will be made public to the community on a wiki afterwards.

Who should attend the workshop? Everyone of course! Experienced agilists will contribute to the content of the patterns. Those who are practicing but still new to agile practices will get a chance to reinforce their current adoption of practices. Finally those who are still considering agile practices will make sure the patterns address the current questions they need answered to make more informed decisions which practices to adopt and how to do so.

2.4.2 Biography

Amr Elssamadisy has been working professionally as a software developer, architect, manager, consultant, etc. (too many titles ☺) for over 12 years helping build software systems. His first agile development project was a large project XP effort in 1999 where he had a chance to work and learn from some of the best in the field. Since then he has lead, participated, and guided teams in several large and small agile development projects. He is currently a Principal Consultant at Valtech Technologies where he helps clients solve their problems in building software development systems with different technologies and practices.

Ahmed Elshamy serves ThoughtWorks, Inc. as a Senior Software Developer in the creation of large-scale distributed object applications. During his tenure at ThoughtWorks, Ahmed has worked on multiple strategic agile projects since 2000 in both leasing and insurance domains. He mentored and enabled multiple teams in applying agile software development process and agile best practices. He also worked on integration projects that integrate client server application with legacy systems. Ahmed has experience in .Net and Java technologies. Ahmed contributed to ChiliPlop 2006 hot topics.

2.5 Human & Social Factors in Software Engineering: Motivation and De-Motivation in Agile Development

Helen Sharp
Department of Computing at The Open University & City University London, United Kingdom

Tracy Hall
University of Hertfordshire, United Kingdom

Bjørnar Tessem
University of Bergen, Norway

Frank Maurer
University of Calgary, Canada

Daniel Karlström
Lund University, Sweden

Yvonne Dittrich
IT-University of Copenhagen, Denmark

2.5.1 Introduction

What do you enjoy about developing software? What would make you leave an agile project and join a non-agile project? What kind of people do you like to work with? When does software development become a drag? There is some evidence that job satisfaction increases when using agile methods – why?

In this workshop, we will use an interactive format to answer these questions and explore the effect of organisational, technical and social context on developer motivation. Please come along and share your experiences, pain, joy and insights about what makes software engineering a rewarding (or a frustrating?) occupation.

This workshop is suitable for anyone who is interested in motivation in software engineering, as a developer, a manager or a researcher.

2.5.2 Biography

Helen Sharp is a Senior Lecturer in the Department of Computing at The Open University and a Senior Visiting Fellow at City University London. She is very active in the agile and interaction design communities and has had a long association with the SPA series of interactive conferences in the UK. Her research focuses on understanding software development practice in order to integrate user-centred concerns and software engineering, and to provide suitable support for software engineers.

Tracy Hall leads the Systems and Software Research Group at the University of Hertfordshire. She is an international expert in the human and organisational aspects of software development processes. She specialises in the empirical investigation of non-technical issues within software engineering. She has published widely in these areas. She is an Associate Editor of the Software Quality Journal, a programme committee member for both the IEEE International Metrics Symposium and the ICSE Experience track. She also co-chaired the Professional Awareness in Software Engineering Conference.

Dr. Bjørnar Tessem is head of the PAILab (programming and artificial intelligence) research group at the University of Bergen, Norway. His research interests include qualitative studies of software engineering, agile methods, intelligent tools for software engineering and artificial intelligence in general. He has chaired and co-chaired several international and national conferences on a wide range of ICT topics.

Dr. Frank Maurer is the head of the e-Business engineering (ebe) group at the University of Calgary. His research interests are agile software methodologies, web engineering, globally distributed software development, experience management and integrating agile methods and interaction design.

He is a member of the PC of the ICSE 2005 Experience Report section and also responsible for the ICSE 2005 proceedings. He was program chair of SEKE 2004 and program co-chair of XP Agile Universe 2003. He is and was a program committee member of various conferences in the area of software engineering, agile methods and knowledge management.

Dr. Daniel Karlström is a member of the Software Engineering Research Group at Lund University. His research interests include integrating agile methods in various organizational environments, human aspects of software process improvement and qualitative and quantitative methods in software engineering research. He is poster co-chair at XP2006.

Dr. Yvonne Dittrich works an associate professor the IT-University of Copenhagen. Her research interests are use oriented design and development of software, end especially the flexibilization of software products and processes in order to accommodate the co-development of work practices and technology. She developed an empirical research approach Cooperative Method Development together with industrial partners which is based on problem oriented software process improvement as a learning cycle both for the industrial partner and for the researchers involved. This approach has been applied in a number of research projects in cooperation with industry.

2.6 User Stories for Agile Requirements

Mike Cohn
Mountain Goat Software, Colorado, U.S.A.

2.6.1 Introduction

Software requirements is a communication problem. Those who want the new software (either to use or to sell) must communicate with those who will build the new software. To succeed, a project relies on information from the heads of very different people: on one side are customers and users and sometimes analysts, domain experts and others who view the software from a business or organizational perspective; on the other side is the technical team.

If either side dominates these communications, the project loses. When the business side dominates, it mandates functionality and dates with little concern that the developers can meet both objectives, or whether the developers understand exactly what is needed. When the developers dominate the communications, technical jargon replaces the language of the business and the developers lose the opportunity to learn what is needed by listening.

What we need is a way to work together so that neither side dominates and so that the emotionally-fraught and political issue of resource allocation becomes a shared problem. Projects fail when the problem of resource allocation falls entirely on one side. If the developers shoulder the problem (usually in the form of being told "I don't care how you do it but do it all by June") they may trade quality for additional features, may only partially implement a feature, or may solely make any of a number of decisions in which the customers and users should participate. When customers and users shoulder the burden of resource allocation, we usually see a lengthy series of discussions at the start of a project during which features are progressively removed from the project. Then, when the software is eventually delivered, it has even less functionality than the reduced set that was identified.

By now we've learned that we cannot perfectly predict a software development project. As users see early versions of the software, they come up with new ideas and their opinions change. Because of the intangibility of software, most developers have a notoriously difficult time estimating how long things will take. Because of these and other factors we cannot lay out a perfect PERT chart showing everything that must be done on a project. So, what do we do?

We make decisions based on the information we have at hand. And we do it often. Rather than making one all-encompassing set of decisions at the outset of a project, we spread the decision-making across the duration of the project. To do this we make sure we have a process that gets us information as early and often as possible. And this is where user stories come in.

A user story describes functionality that will be valuable to either a user or purchaser of a system or software. User stories are composed of three aspects:

1. a written description of the story used for planning and as a reminder
2. conversations about the story that serve to flesh out the details of the story
3. tests that convey and document details and that can be used to determine when a story is complete.

Because user story descriptions are traditionally hand-written on paper note cards, Ron Jeffries has named these three aspects with the wonderful alliteration of Card, Conversation, and Confirmation [1]. The Card may be the most visible manifestation of a user story, but it is not the most important. Rachel Davies [2] has said that cards "represent customer requirements rather than document them". This is the perfect way to think about user stories: While the card may contain the text of the story, the details are worked out in the Conversation and recorded in the Confirmation.

There are a number of reasons why projects benefit from using user stories as their requirements technique. A few are worth highlighting briefly here. First, user stories shift the emphasis from documents to talking and from writing to understanding. Writing things down does have advantages: written words help overcome the limitations of short-term memory, distractions, and interruptions. But, so many sources of confusion – whether from the imprecision of written words or from words with multiple meanings – go away if we shift the focus from writing requirements down to talking about them.

Our goal with user stories is not to document every last detail about a desired feature; rather, it is to write down a few short placeholding sentences that will remind developers and customers to hold future conversations. Many of my conversations occur through email and I couldn't possibly do my job without it. I send and receive hundreds of emails every day. But, when I need to talk to someone about something complicated, I invariably pick up the phone or walk to the person's office or workspace.

Second, user stories have the advantage of being easily understood by project developers and customers. An approach such as use cases often requires that customers (and often developers) be trained on how to read a completed use case template with its preconditions, post-conditions, extensions, and so on. This places developers and customers on unequal footing in the communication that must occur between those who want the software and those who will develop it. Because users find user stories more immediately comprehensible they are more likely to become fully engaged participants in the design of their software.

The final advantage to user stories I'll mention here is that stories encourage deferring detail. It is very common for a team to create an initial user story that is quite large, perhaps representing many months of work. This initial large story will need to be disaggregated by the team into many smaller stories, but the work of disaggregating the story is deferred until the team is nearly ready to work on the smaller stories. This just-in-time approach to requirements provides the dual benefit of potentially avoiding discussions about requirements that are subsequently removed from the project and causing the conversations about the user's need to occur when it is most valuable – right when the team begins to work on it.

The technique of expressing requirements as user stories is one of the most broadly applicable techniques introduced by Extreme Programming. User stories are an effective approach on all time constrained projects, not just those using XP. In this session we will look at how to identify and write good user stories. The session will describe the six attributes all good stories must exhibit and present thirteen guidelines for writing better stories. We will explore how user role modeling can help when gathering a project's initial stories.

2.6.2 Biography

Mike Cohn is the founder of Mountain Goat Software, a process and project management consultancy and training firm. He is the author of *Agile Estimating and Planning* and *User Stories Applied for Agile Software Development*, as well as books on Java and C++ programming. With more than 20 years of experience, Mike has previously been a technology executive in companies of various sizes, from startup to Fortune 40. A frequent magazine contributor and conference speaker, Mike is a founding member of the Agile Alliance, and serves on its board of directors. He can be reached at mike@mountaingoatsoftware.com.

2.6.3 References

[1] Jeffries, Ron. 2001. Essential XP: Card, Conversation, and Confirmation. XP Magazine (August 30, 2001).

[2] Davies, Rachel. 2001. The Power of Stories. XP 2001. Sardinia.

2.7 Value Stream Mapping

Mary Poppendieck & Tom Poppendieck

2.7.1 Introduction

Value Stream Mapping is a Lean tool that has a long history of uncovering waste in manufacturing and operational settings; it is also a great tool for software development. In this session, participants will learn simple rules for creating value stream maps, and teams will create maps of real situations. The resulting value stream maps will be presented and critiqued, so participants can envision for themselves how they might use this practical tool.

Participants will discover, through hands-on experience and discussion, how to create and use value stream maps. As the maps are constructed and analyzed, participants will discover how value stream maps create a new perspective on the software development process, one that they can use to evaluate their workflow and pinpoint the biggest opportunities for improvement.

This workshop will interest managers and team leads who understand Agile development and who are looking for a practical tool they can use to analyze and improve their workflow.

2.7.2 Biography

Mary Poppendieck has been in the Information Technology industry for thirty years. She has managed solutions for companies in several disciplines, including supply chain management, manufacturing systems, and digital media. As a seasoned leader in both operations and new product development, she provides a business perspective to software development problems.

A popular writer and speaker, Mary's classes on managing software development have been popular with both large and small companies. She is co-author of the book Lean Software Development: An Agile Toolkit, published by Addison Wesley in May, 2003 and winner of the Software Development Productivity Award in 2004.

Tom Poppendieck is a principle in Poppendieck.LLC, assisting organizations improve their software development capabilities. He has held positions of IT Architect, Supervisor, and VP of Product Development at various Twin Cities companies. Prior to that he was a Physics Professor and Director of Academic Computing at Hamline University. He is co-author of Lean Software Development: An Agile Toolkit, which won the Software Development Magazine Productivity Award in 2004 for bringing lean principles to software development.

2.8 How to Deliver on the Value Proposition from Real Life XP Projects

Jan-Erik Sandberg
Chief Quality Officer, Objectware, Norway

Lars Arne Skår
Chief Technology Officer, TietoEnator Banking Solutions

2.8.1 Introduction

Although XP is no longer as controversial as it used to be, it is still hard to initiate and deliver on XP projects in a way that delivers on the promised values. It is as hard as ever to maintain a focus on the XP practices in projects over time. This may be in areas such as:

- Contractual agreements.
- Old habits that die hard (both from developers and customers).
- Accepting that significant start-up investments are needed.
- Getting true acceptance of the practices – from the tangible test and build techniques to softer parts like refactoring and pair programming, which still can be very controversial.
- How to interface to formal "toll gates" in larger organizations.

The organizers have been through multiple large scale successful agile projects, which have delivered the starting and ongoing values of an XP approach. Most of the initial customers and organizations were used to traditional waterfall and similar approaches.

The workshop will consist of initial group discussions to share experiences and opinions, and a final fishbowl discussion with a focus on what values XP projects actually provide and how to communicate those to the stakeholders.

The organizers wish to gather project managers, developers and ideally potential customers of xp/agile projects to discuss and share ideas on what values XP projects actually provides for customers and managers. In order to get new views on the topic, a group with mixed experience level on XP is desirable.

2.8.2 Biography

Jan-Erik Sandberg, Chief Quality Officer, Objectware, Norway Jan-Erik Sandberg founded the Norwegian eXtreme Programming forum 5 years ago, with a strong dedication to evangelize extreme programming to the norwegian community. He has presented XP on several international events – at TechED Europe 2004 he was rated one of the "Top Speakers". For several years, he has been in charge of the agile track at one of Scandinavia's largest software conferences. Currently he is technical project manager for an agile project with over 45 developers, and a timeframe of 8 years.

Lars Arne Skår, Chief Technology Officer, TietoEnator Banking Solutions Lars Skar joined the Norwegian eXtreme Programming forum 3 years ago, and shares Jan-Eriks ambition to evangelize extreme programming and agile development in the Norwegian system development community. Lars actively participates at OOPSLA and other international conferences as conference speaker and in workshops. During his 16 years of experience in the consultancy industry, he has always had a strong focus on training and coaching others. In addition to act as a coach, he has also been an instructor on several international training events.

2.9 The Coders' Dojo

Christophe Thibaut & Emmanuel Gaillot
Octo Technology, France

2.9.1 Introduction

If I want to learn Judo, I will enroll at the nearest dojo, and show up for one hour every week for the next couple years. After two years I may decide to start studying in earnest to progress in the art. Years of further training might be rewarded with a black belt, which merely signals a different stage of learning. No master ever stops learning. If I want to learn object programming... my employer will pack me off to a three-day Java course picked from this year's issue of a big training firm's catalog. Nuts to that – acquiring coding skills is not an "instant gratification" process. We propose a more appropriate way of teaching and learning programming, respecting the depth and subtlety of the craft.

The Coders' Dojo in Paris has been running on a weekly basis since January 2005. A "sister" Dojo is also running bi-monthly basis in Finland. The session itself has been presented at XP2005, XP Day Benelux 2005 and XP Day Germany 2005.

The Dojo is a long-term process; this one-time session encourages participants to start their own. The format is between a tutorial (interactive instruction) and a workshop (instructive interaction). We pay attention both to participants' payback in improvement of their programming and TDD skills, and to ideas for improving the Dojo format.

The Coders' Dojo is a weekly programming class. Programmers of varying skill levels meet as equals. They come together – in physical, not virtual space – around an ongoing series of coding challenges, usually small in scope, often patterned after "pragmatic" Dave Thomas' idea of "coding Kata". The session replicates a "typical" session of the Coders' Dojo.

The Dojo uses different formats according to participants' inclination. In the session we will use the Randori format, in which:

- at any one time there is one pair "on the mat" (i.e. at the keyboard)
- the rest of the group observes via beamer
- the pair writes a test, codes, and refactors for one 5-minute timebox
- at the end of the timebox, the copilot becomes driver, a new copilot comes forward from the group of observers, and the driver goes back to observing.

The session schedule is as follows:

09:00–09:10	Bootstrapping: introduction to the Coders' Dojo, rules of engagement
09:15–09:30	Brainstorm possible topics, vote
09:30–10:15	Randori – first topic
10:15–10:30	Debrief: what worked, what needs improving, new ideas
10:30–10:45	Break
10:45–11:30	Randori – second topic (or continue first)
11:30–12:00	Session retrospective

2.9.2 Biography

Christophe Thibaut (cthibaut@octo.com) has been working in IT for 15 years as a Developer, Project Manager and Consultant for Banking and Industrial groups in France. He is interested in everything that can help teams in their day to day work of creating liable, usable and maintainable software. As such he has been practicing Extreme Programming with small teams since 2001 in the context of lecacy applications. He's also involved in the Developers Dojo Experiment in Paris since it's creation.

Emmanuel Gaillot (egaillot@octo.com) is a software engineer and an experienced designer for theatre and dance. He has adapted XP practices and principles to the theatrical production process, and he currently works on instilling theatre practices back into the field of software making. Emmanuel's areas of expertise and interests include self-organizing teams, software making and Extreme Programming. He is involved in the conduct of the Coders' Dojo Experiment in Paris, France, where he also works for Octo Technology as an XP coach.

2.10 Mapping XP

Kent Beck & Cindee Andres

2.10.1 Introduction

We will explore the primary practices of XP in detail using mind mapping exercises. You will examine your needs and find practices to address them. We will discuss the change process, how to reach agreement on goals and principles, how to implement new practices and how to sustain them. You will make a plan to share with your team and set up incentives for accountability. This tutorial will be interactive and involve many colored felt pens.

2.10.2 Biography

Ms. Andres is a co-author of Extreme Programming Explained: Embrace Change, 2nd Edition. Her professional interests include team and individual psychology and facilitating change with large-scale transformative conversations. She holds a B.A. in Psychology from Pacific Union College with advanced work in Women's Studies at the University of California at Santa Cruz and Psychology at Portland State University.

Kent Beck is the founder and director of Three Rivers Institute and an Agitar Software Fellow. His career has combined the practice of software development with reflection, innovation, and communication. His contributions to software development include patterns for software, the rediscovery of test-first programming, the xUnit family of developer testing tools, and Extreme Programming. He currently divides his time between writing, programming, and coaching. Beck is the author/co-author of Extreme Programming Explained: Embrace Change 2nd Edition, Contributing to Eclipse, Test-Driven Development: By Example, Planning Extreme Programming, The Smalltalk Best Practice Patterns, and the JUnit Pocket Guide and the forthcoming Implementation Patterns. He received his B.S. and M.S. in Computer Science from the University of Oregon.

2.11 Agile Process Anti-patterns

Wayne Allen
Corillian Corporation, U.S.A.

2.11.1 Introduction

Agile methodologies are just as prone to mistakes, deviations and subversion as any other methodology. This workshop is intended to discover and share agile process anti-patterns then discuss and present creative solutions.

Anti-patterns, also referred to as pitfalls, are classes of commonly-reinvented bad solutions to problems. They are studied, as a category, in order that they may be avoided in the future, and that instances of them may be recognized when investigating non-working systems.

This workshop focuses on participants' (less than perfect) experiences in the workplace and mines the collective wisdom of the group to come up with creative solutions via a fishbowl process followed by breakout sessions and then group presentations. This is a half day workshop.

A fishbowl is structured conversation technique where a small number of people sit in front of the group (in the fishbowl) and converse with each other. The audience listens and at any time a member of the audience may become a member of the fishbowl, replacing one of the existing members.

Sample starter agile process anti-patterns: Testing Not Completed by Iteration End, Functional Specialists & Handoffs, Lack of Automated Testing, 100% Utilization.

This workshop is for anyone who has ever worked on an imperfect agile team or who desires to understand how things can go wrong. Participants will come away with a broader understanding of the ways agile projects can get off track and a variety of solutions for addressing these problems.

2.11.2 Biography

Wayne Allen has been a technology professional since 1987 and is currently a Product Development Manager for Corillian Corporation where he manages several agile teams. He has been coaching organizations in adoption of agile methods since 2000. Starting in 2002 Wayne began specializing on the areas outside the core of agile development, namely product management, quality assurance, human factors, documentation and project management. Wayne is one of the organizers for the Portland, OR USA XP Users Group (XPDX http://www.xpdx.org).

2.12 Exploring Agile Project Parameters: How to Implement Agile Development on Real Projects

Rachel Davies
Agile Experience Limited, United Kingdom

David Hussman
SGF Software, U.S.A.

2.12.1 Introduction

When a team sets out to apply agile software development on a project, they soon will discover there are a bunch of parameters and variables that need to be set. For example, how long will their development cycles be, what tasks will the apply pair programming to, who will maintain tracking information, etc.

This workshop aims to provide a guide to teams on topics that they will need to develop working agreements on within their project teams to help the team get clear about how they want to apply agile techniques. The goal of this workshop is to produce a set of questions that teams can use at the start of a software development project to help them get clear about how they plan to apply agile techniques.

This session will interest coaches, team leads and managers who want to set their project teams up for successful implementation of agile software development. Participants should have some experience of working in agile software development.

2.12.2 Biography

Rachel Davies is as an independent coach and facilitator in UK. She is passionate about agile software development because it can increase the chance of success in the face of complex problems and recognizes that teams are made-up of individuals rather than resources. Rachel specializes in XP and Scrum flavours of agile development and advocates the use of retrospectives to help teams adapt their process to their context. Rachel is a frequent presenter at agile conferences and serving director of the Agile Alliance. Rachel is founder of Agile Experience Limited and also a senior consultant with Cutter Consortium.

David Hussman has designed and created software for more than 13 years in a variety of domains: digital audio, digital biometrics, medical, retail, and education to name a few. For the past 6 years, David has mentored and coached agile teams in the U.S., Canada, Russia, and Ukraine. Along with leading workshops and presenting at conferences in the U.S. and Europe, David has contributed to numerous publications and several books. David co-owns the Minneapolis based SGF Software, is a senior consultant with Cutter Consortium, and has contributed to the agile curriculum for Capella University and University of Minnesota.

2.13 Extreme Construction: Making Agile Accessible

Joe Bergin & Ivan G. Seidenberg
School of Computer Science and Information Systems of Pace University, U.S.A.

Jutta Eckstein
IT Communication, Germany

Fred Grossman
Ivan G. Seidenberg School of Computer Science and Information Systems of Pace University, U.S.A.

2.13.1 Introduction

Extreme Construction is an active learning exercise in which teams of participants experience the basics of Extreme Programming (XP) in a day, without programming. This workshop uses non-programming construction techniques to teach most of the XP practices in a fun and interesting format. The teams build a non-software artifact using a variation on XP. They see how the synergy of the XP practices brings benefits to a software development organization.

Many people have an interest in agile methodologies, and XP in particular. Managers need a way to evaluate XP from an institutional benefits perspective and knowing how it actually works can be a help in this. Programmers need to learn the practices before they can begin to effectively use them. They also need a way to see how their current development habits may be in conflict with XP practices. Educators want effective ways to introduce XP into their courses. This exercise provides required background for all of these needs in a fun and informative way.

The basic idea is that teams use XP practices to build a physical artifact out of basic arts and crafts materials: construction paper, pipe cleaners, clay, glue, etc. Some artifacts are inherently more complex than others. The simple ones have lots of loosely connected parts: playground, town square, etc. In the more difficult ones, the parts are intricately connected, such as a two-masted sailing ship.

Participants need no particular background to benefit from this exercise. It is tailored to the audience, of course, but if most of the participants have no knowledge whatever of XP, then the day begins with a one-hour overview of XP values, principles and practices. This gives a managerial/stakeholder view of XP and its purported benefits as well as a quick introduction to its practices. If most have read some background material, such as Kent Beck's *Extreme Programming Explained*, then this portion is omitted. This might be the case if the exercise was to be given to developers who have the overall view, or educators who have done some reading, but need the hands-on experience. We usually include this, however, as the nature of the exercise makes it most useful to non-programmers, such as managers and project owners.

The second part of the day is the game itself. Participants are divided up into teams of about ten. There are three roles with at least two people in each role: customer, developer, and tracker. We use two customers rather than one to get the creative juices flowing between them. Note that we also have an exercise, Pair Story Writing, which dramatically demonstrates the effect of pairing on creativity. The trackers keep time and watch for things like coercion between developers and customers. While one coach takes the customers aside to choose the artifact to be built and begin story writing, the other works with everyone else to reinforce the rules of the game, which are generally, simply the practices of XP, pairing, test first building, working with the customer to improve story writing, estimating stories, and the like. Then the participants come together (Whole Team) to build. Developers estimate the stories (Planning Game) and give a velocity (in minutes). The coaches set the length of the iteration (twenty or so minutes). The customers choose stories and the developers then build them. Sketches are used for testing (Test Driven Development). All building is done in pairs (Pair "Programming"). Meanwhile, the trackers assure that the customers don't build or estimate, for example, and the developers don't decide "what" to build. Several iterations are done, with three being about the right number to get the flavor and let some problems appear that will be discussed in the retrospective.

We conclude the exercise with a retrospective. In many ways this is the most important part of the day. The participants never perform the practices perfectly, of course. This is their first attempt at it. If they have some background in software then they will most naturally fall back on familiar practices. This usually results in non-optimal progress. The coaches of the exercise watch for problems as the game proceeds and especially note where problems result from not following the practices. The retrospective brings this out. We have seen lots of situations in which the development would have been much smoother but for old assumptions and practices coming into play. For example, one team had a lot of trouble because one of the "manager" types wanted to control the planning game phase, and it became a bottle-neck and so they ran out of time with few stories properly estimated. We have seen teams that really wanted to do a lot of up-front design or made assumptions about the needs of the customer without customer agreement, both resulting in discarded artifacts.

Not only do the teams learn the practices, both on an intellectual level and on a more hands-on, visceral, level, but we have found that when this is used by a real team just prior to beginning a real project, that the teams come together quickly. The level of trust required to perform communication-intense development in the XP way begins to develop, partly because the exercise is fun as well as informative.

This workshop has been given many times in industrial, conference, and academic settings. It has proven to be an effective way to train individuals and teams, and also to help a new agile team coalesce into an effective working unit.

2.13.2 Biography

Joe Bergin (csis.pace.edu/~bergin, jbergin@pace.edu) is an educator and practitioner with 33 years experience, 17 in OO, and 5 or so in agile practice and consulting. He is a frequent participant at agile development conferences. He has, with Fred, co-led workshops at OOPSLA and CASCON on XP, including in the past two years. He uses XP in the classroom and developed this exercise as an outgrowth of the Extreme Hour, needing something that covered more of the practices, but still possible to do in a day. He is the author of four textbooks on object-oriented programming. In addition he is an active pattern writer, with many papers (co-authored) on pedagogical patterns. One current project is a pattern language for agile development, including all of the well-known advice as well as solutions to some common pitfalls. Effective teaching is one of his life goals and he believes in agile approaches to achieve this. The Extreme Construction exercise is one result as it moves the student forward on multiple fronts simultaneously.

Jutta Eckstein (www.jeckstein.com, info@jeckstein.com) is an independent consultant and trainer for over ten years. She has a unique experience in applying agile processes within medium-sized to large mission-critical projects. This is also the topic of her book *Agile Software Development in the Large*. Besides engineering software she has been designing and teaching OT courses in industry. Having completed a course of teacher training and led many 'train the trainer' programs in industry, she focuses also on techniques which help teach OT and is a main lead in the pedagogical patterns project. She has presented work in her main areas at ACCU (UK), OOPSLA (USA), OT (UK), XP (Italy and Germany) and XP and Agile Universe (USA). She is a member of the board of the AgileAlliance and a member of the program committee of many different European and American conferences in the area of agile development, object-orientation and patterns.

Fred Grossman (grossman@pace.edu) has been teaching for more than 30 years and has been involved in software development for 40 years. He is a professor and Program Chair of the Doctor of Professional Studies in Computing in the Ivan G. Seidenberg School of Computer Science and Information Systems of Pace University. He has, with Joe, co-led workshops at OOPSLA and CASCON on XP and trained and coached XP teams in academic and industrial settings.

3. Activities

3.1 You got Agile – How do you Convince the Rest

Erik Lundh
Compelcon AB, Helsingborg, Sweden

3.1.1 Introduction

Key Question: How do I sell XP/Agile to “other people”, making it possible to start or spread XP/Agile within my organization?

Invited:

- Developers, managers and business stakeholders, the people who want to introduce and implement XP or other agile approaches within their company
- XP Customers who need better support from their company
- Already working XP/Agile teams who need to convince the rest of the company to go Agile
- And all people who have successfully convinced the rest to go Agile.

Participants will share challenges and questions, success stories, strategies and tactics as well as actual experience of how to convince companies to go agile.

What we should cover

- Getting People Interested

  How do we tell the success stories in figures to non-XP-ers, i.e. how going Agile pays off? We look at some examples of companies going Agile successfully, in words that appeal to our different stakeholders. And we look at how Lean Software Development helped us explain why XP and Agile works.

- Strategies for Change

  We look into a few different approaches to introduce process improvement such as XP and other agile methods. What works and what usually fails.

- How We Changed These Companies

  How do we explain agile to different stakeholders and have them turn over and tell others the same story?

  What matters for developers, customers, and management?

  How we

  - enable communication, and deal with die-hards.
  - position XP as an experience that bootstraps "the learning organization".
  - convince the business people with simulations and games.

- How to Bring People On Board with Simulations
  - How to prepare a simulation.
  - How to sell it to business people.
  - Why do simulations work so well?
  - Things to look out for.
  - What to count on.
  - Bigger groups – the audience is the market.

3.1.2 Biography

Erik Lundh: C*hange agent with 25 years in the software industry. Technical, business and peopleware person.*

Some people at Lund University brought XP to Erik's attention, back when Erik was working with cross-industrial R&D centers for products with software. Erik recognized key players like Beck and Cunningham from the patterns movement and went to XP2000, spoke about XP in Sweden, started workgroups, organized a 200-head Swedish process improvement conference with an XP theme, and finally found himself deeply involved in XP/Agile both in Sweden and internationally.

Erik has been invited to several panels with Beck, Jeffries and other well-known XP profiles and is regarded as one of the most experienced XP coaches in Sweden. Erik currently coaches successful XP-teams, gives well-attended XP seminars in major cities of Sweden, is sometimes honored to be a host of guest appearances by profiles like Ward Cunningham and Mary Poppendieck, and tries to find time to finish writing his big book on XP with practical advice from a seasoned XP coach. Erik started to use XP and XP coaching as experience-based process improvement in 2000. His teams are successful with XP and usually get company-wide recognition.

In 2004, Erik became CSM through Ken Schwaber's excellent Scrum Master class, only to find that he always has done Scrum as part of his approach to XP.

Erik is a board member of SPIN-Sweden, and an involved sponsor in most, if not all, of the Swedish SPIN-chapters. His local chapter SPIN-SYD is the largest in Sweden, with over 40 companies including Ericsson and ABB. SPIN-SYD and Lund University was key to Erik's early work in XP and Agile. Lund University has continued independently and introduced full XP as the first project method to over 500 students over 5 years.

Erik currently lives with his wife and two kids in Helsingborg, Sweden, just across the water from the Danish castle of Kronborg, home of Shakespeare's Prince Hamlet.

3.2 Agile GUIs – Test Driving Your S-Wing

Patrick Kua
ThoughtWorks Inc, United Kingdom

3.2.1 Introduction

Client side applications are notoriously difficult to test, with many of the traditional approaches leveraging screen scraping techniques resulting in both brittle and costly tests. There are a number of alternative ways of testing Swing GUIs, and we will be looking at one of these that make testing Swing applications easier and more reliable.

We will investigate how you can apply this testing technique for TDDing an entire Swing application that will be both robust and reliable. We will look at how you write acceptance level tests and the patterns that you could implement in a swing application that allows you to fully test all the components before you get an acceptance test passing.

We will demonstrate the robustness of tests by demonstrating how you can changing the way the components are laid out without breaking tests. We will also discuss all the difficulties, tips and tricks that you should be aware when you do decide to write tests for your application.

Attendees should be strong Java developers who have been exposed to unit testing and open to TDD. Any budding TDDers are welcome where you will learn how to better express requirements in tests, and learn how to effectively TDD swing applications, avoiding the painful way of brittle UI testing.

You will learn how to leverage the Model View Presenter for structuring an application that is almost fully testable without having to actually visually render any components. You will learn how to write acceptance tests that read well and that are can be easily refactored to use different libraries.

3.2.2 Biography

Patrick Kua is a budding agile coach and hardcore agile developer who is currently working for Thoughtworks, a software development consultancy. He is driven by lightweight methods of delivery that means his team gets to write more code more often that delivers greater value. He is a big believer in self-empowered teams and achieves this by continually spreading the word about better ways of writing software. He believes that Test Driven Development is an effective technique for writing Just The Right Amount of Code that is both more understandable and more maintainable.

3.3 Pair Storytelling – "Once upon a time there were two…"

Fred Grossman & Joe Bergin
Ivan G. Seidenberg School of Computer Science and Information Systems of Pace University, U.S.A.

3.3.1 Introduction

Pair programming is one of the key practices of eXtreme Programming (XP). Even though research has shown that working in pairs is more productive than working alone, many people are skeptical about the practice until they try it. Managers are often wary of the practice, believing it will increase costs; developers are wary, believing it will curb creativity. The Pair Storytelling activity can provide an experience to address these concerns in a fun and informative manner. It is normally used with a team that is just in the formative stage that is about to begin an agile project, usually for the first time. We include managers as well as business-side stakeholders in our training whenever possible.

Participants experience the creative value and personal satisfaction of pairing. This workshop activity uses non-programming story-writing techniques to show novices, experienced developers, managers, and even non-technical people, what it is like to work in pairs to produce something creative.

Pair storytelling involves writing a story that is partially specified by another person (the customer/product owner). The stories used in this activity are based on familiar children's stories. The task is to update them in a "customer" specified way, being as creative as you like. This is done first as individual work and then with pairing. A few samples of individual and paired stories are read to the group. At the end of the activity, a brief retrospective permits the group to discuss the experience.

More specifically, the participants first write something that is partially specified. While many different things could be used as the basis, we normally use common fairy tales such as can be found at http://surlalunefairytales.com, for example, Hansel and Gretel, Jack and the Beanstalk, and Goldilocks and the

Three Bears. The assignment is to choose one of these and update it to the 21st century, or give it a high tech component. We provide the names and a brief synopsis of the stories, and each individual chooses one and works on it for twenty minutes or so. If time is short and the environment allows for prior work, we sometimes give this as an assignment to participants the day before the training session and use the result as a ticket to the session itself. We typically schedule 60 to 90 minutes for the exercise and 30 minutes for the retrospective. Each writing iteration lasts 20 to 30 minutes, with short readings ("acceptance tests") in between.

The stories we use are fairly commonly known, but all have a western (usually Germanic) basis, and so there are some cultural dependencies that you can overcome in an international group. Most cultures have such stories, we expect, and participants can choose one known to them if none of the standard ones are familiar.

After the initial writing period, a few of the stories are read to get a sense of what was done. Then the participants are paired, asked to choose a different story than either of them previously worked on. The assignment is the same, but we often add "as creatively as you like". In contrast to the first part, the room now gets a bit noisy and people sound happier – as if they are enjoying themselves. If the pairs are working at a computer, then naturally one of them "drives" (controls the keyboard) while the other "navigates" providing suggestions. There are quite a lot of suggestions and interplay in nearly all cases and very little disagreement. This interplay seems to increase the creative flow and the stories produced are in many ways both better and more interesting. We also notice that the work is truly "paired". We have never seen a case of division of labor, though it is admittedly difficult to imagine what such a division might be here. In any case it sets the tone for paired programming in a natural way.

We have the participants read as many of the new stories as we have time for and often ask to publish the stories if a wiki is available. One of the creators of this activity, being quite deaf, has a lot of trouble with this reading part, as there is so much laughter in the room that it is difficult to hear.

The session closes with a short retrospective in which we try to bring out both the increase in creativity that occurred as well as the team/trust building aspects of the exercise. Normally our participants know little of one another when we do this. During the retrospective we ask the usual questions ("What was this like for you?"), as well as explore with them whether it made any difference in their perceptions of what paired programming might be like.

While this exercise is especially useful for those without programming skill and who therefore can't do paired programming, it is also useful in the team formation stage of a group of developers. If we were to use pair programming initially, there would be a number of psychological barriers that don't appear here. For example, the perception, real or not, of pairing across different programming skill levels is a common objection to the practice of pairing itself. The fact that this exercise is very non-threatening means that ego is reduced and a level of personal trust can develop even among those of different levels of skill and experience.

Fred Grossman initiated this exercise in a course jointly taught with Joe Bergin. Together they have used it several times and have extended it into its present form. It is one of the techniques that they use to mold a group of strangers into an effective team quickly so that the members not only work efficiently together, but learn to care for one another as well, building personal commitment to the group and its work.

3.3.2 Biography

Fred Grossman (grossman@pace.edu) has been teaching for more than 30 years and has been involved in software development for 40 years. He is a professor and Program Chair of the Doctor of Professional Studies in Computing in the Ivan G. Seidenberg School of Computer Science and Information Systems of Pace University. He has, with Joe, co-led workshops at OOPSLA and CASCON on XP and trained and coached XP teams in academic and industrial settings. Fred initiated this exercise as a way to demonstrate the creativity of pairing in a context open to non-programmers.

Joe Bergin (csis.pace.edu/~bergin, jbergin@pace.edu) is an educator and practitioner with 33 years experience, 17 in OO, and 5 or so in agile practice and consulting. He is a frequent participant at agile development conferences. He has, with Fred, co-led workshops at OOPSLA and CASCON on XP, including in the past two years. He uses XP in the classroom and worked with Fred to develop this exercise into its present form. He is the author of four textbooks on object-oriented programming. In addition he is an active pattern writer, with many papers (co-authored) on pedagogical patterns. One current project is a pattern language for agile development, including all of the well-known advice as well as solutions to some common pitfalls. Effective teaching is one of his life goals and he believes in agile approaches to achieve this. The Pair Storytelling exercise is one result as it is effective both in showing the value of a technique, and also in team building.

3.4 Open Space Event

Charlie Poole
Poole Consulting L.L.C., U.S.A.

3.4.1 Introduction

Over the last 20 or more years, the Open Space movement has grown to the point where entire conferences based on its principles are not uncommon. Working without an agenda, but with specific strategies to encourage and direct participation, groups of up to 1000 people have successfully self-organized around a complex problem, held meetings and published results.

Although XP2006 is organized as a traditional conference, with sessions and speakers scheduled in advance, it also incorporates an Open Space "track", which will allow participants to create and attend sessions around topics of their own choosing, even topics that arise right at the conference itself.

The opening session of Open Space takes place as a full-conference session on the first day. At that time, the "rules" of Open Space will be explained and individuals will be invited to present their session proposals very briefly. As each session is presented, the sponsor will select a date and place using a large wall-display and will place the notice of the session in the appropriate location.

Sessions may be on any subject that relates to the conference. They may be in the form of a short demonstration, a workshop, a discussion or anything else that the presenter wishes. It’s not even necessary for the presenter to be an expert in the subject of the session. In fact, sponsoring a session on some topic, about which you would like to know more, is an excellent way to tailor the conference to your own needs!

A special feature of the Open Space at XP2006 is that unregistered spouses and partners of conference attendees are also invited to sponsor sessions.

In order to get things started, we asked several individuals to commit in advance to Open Space sessions. These sessions, which will be presented by Joe Bergin and Fred Grossman (“Pair Storytelling” activity), Patrick Kua (“Test Driving Your S-Wing” activity) and Erik Lundh (“You got Agile – How do you convince the Rest?”) are listed separately.

3.4.2 Biography

Charlie Poole is an independent coach and consultant who has worked for major US and European companies. He is an active participant in the agile development community and an advocate of Test-Driven Development as well as the broader use of tests as a medium of communication in agile teams. His workshops in TDD, GUI Testing and Communicating with Tests (developed jointly with Ward Cunningham) have been presented in various companies as well as publicly. Charlie is a practitioner of both XP and SCRUM and is one of the developers of the NUnit unit-testing framework for the .Net environment.

3.5 Fishbowl: The Convergence of Agile Software Development Practices

Steven Fraser
QUALCOMM, San Diego, U.S.A.

Charlie Poole
Poole Consulting LLC, WA, U.S.A.

3.5.1 Introduction

Are agile software development practices converging? Are practices becoming more integrated and/or more widely adopted than others? In the early 90s there was a convergence of object-oriented design methodologies – is a similar pattern being repeated within the agile software development community? Several years ago conferences featured debates on the number of practices inherent to XP – or for that matter what constituted XP. Is the Agile community on the verge of converging to standardization or do individual practices retain their individually and evangelists/disciples? Perhaps the question should be: “How are different Agile methods coalescing?” (rather than converging) – in that some methods complement each other – for example XP and SCRUM. Come and share your perspectives and experiences on the nature and results (customer engagement, coaching, methods, processes, project management, certification…) of agile convergence.

3.5.2 Biography

Steven Fraser (Fishbowl Impresario and Co-facilitator) recently (January 2005) joined QUALCOMM’s Learning Centre as a member of senior staff in San Diego, California – with responsibilities for tech transfer and technical learning. From 2002 to 2004 Steven was an independent software consultant on tech transfer and disruptive technologies. Previous to 2002 Steven held a variety of software technology program management roles at Nortel and BNR (Bell-Northern Research) – including: Process Architect, Senior Manager (Disruptive Technology and Global External Research), Advisor (Design Process Engineering), General Chair (Nortel Design Forum), and Software Reuse

Program Prime. In 1994 he spent a year as a Visiting Scientist at the Software Engineering Institute (SEI) collaborating with the "Application of Software Models Project" on the development of team-based domain analysis techniques. Since 1994, Steven has regularly moderated panels at ACM's OOPSLA and other software conferences – serving as OOPSLA panels chair in 2003 and as XP2006's General Chair. Steven holds a Doctorate in Electrical Engineering (software graphics standards validation) from McGill University in Montreal, Canada, an MS in Physics (Queen's University at Kingston), and a BS in Physics and Computer Science (McGill University). Steven is a member of the ACM and IEEE.

Charlie Poole (Fishbowl Co-facilitator) is an independent coach and consultant who has worked for major US and European companies. He is an active participant in the agile development community and an advocate of Test-Driven Development as well as the broader use of tests as a medium of communication in agile teams. His workshops in TDD, GUI Testing and Communicating with Tests (developed jointly with Ward Cunningham) have been presented in various companies as well as publicly. Charlie is a practitioner of both XP and SCRUM and is one of the developers of the NUnit unit-testing framework for the .Net environment.

3.6 The Coding Tournament

Lasse Koskela & Markus Hjort
Reaktor Innovations, Finland

3.6.1 Introduction

The coding tournament is a social event where small teams develop clients for a simple multiplayer game using and learning XP practices. The activity provides a hands-on learning environment by incorporating available mentoring with a meaningful goal and backed by existing infrastructure. The session ends with a game play between the participants' client implementations.

This activity is targeted at practitioners of all levels and knowing XP in and out is not a prerequisite. As a chance to get hands-on coaching from practitioners, this session is a multifaceted learning opportunity suitable for acquiring or honing programming skills and techniques as well as for improving social skills in collaborating with other people.

The importance of direct, hands-on experimentation in a meaningful context and in collaboration with peers on the effectiveness of learning new skills cannot be stressed enough. The Coding Tournament realizes this environment for a group of software developers. The setup facilitates opportunities for participants to take advantage of this environment in exactly the way they feel comfortable with and deem most useful.

The coding tournament is a social event where small teams of participants are presented with a development task around a simple multiplayer game. The participants are encouraged to use and experiment with XP practices such as test-driven development and pair programming, and the organizers provide mentoring and assistance throughout the tournament.

The participating teams are provided with enough freedom regarding technology and the implementation in general as to leave room for personal preferences and to accommodate people with different backgrounds. Specifically, the presented problem must not require in-depth knowledge of the domain or technology. Furthermore, the solution must be achievable within the allotted time. With this in mind, project skeletons are made available for selected technologies in order to avoid spending time on infrastructure setup rather than the problem at hand.

The activity ends with a competitive face-off between the participating teams' implementations. This establishes the meaningful context by giving a concrete goal for the teams. The game helps finish the session on a high note as participants can sit back and watch how their creations' compete against each other, with game events visualized on a projector in real time.

3.6.2 Biography

Lasse Koskela (lasse.koskela@ri.fi) is a methodology specialist at Reaktor Innovations. Lasse entered the IT industry to celebrate the new Millennium and has since held roles varying from development to project management and training to consulting, dipping his toes to sales every once in a while. He started promoting agile methods in Finland in 2002, ramped up the local Agile Seminars in 2005, and is currently working on a book on test-driven development for Manning Publications.

Markus Hjort (markus.hjort@ri.fi) is an software architect at Reaktor Innovations. Markus has been in industry over seven years with experience from various technologies. He has actively participated in process and methodology improvement throughout his career. A certified Scrum Master and a member of Agile Alliance with extensive experience on agile methods, Markus kick-started the Coding Dojo events in Finland during 2005.

3.7 Distributed Design and Development Using Agile Methods and Trac

Tarmo Toikkanen & Teemu Leinonen
Media Lab, University of Art and Design Helsinki, Finland

3.7.1 Introduction

Geographically distributed software development is hard to do properly. There are basically only two known solutions: avoid it, or travel to each location weekly. Nine months ago we started a 2.5 year project with developers in four countries, and no budget for weekly travelling.

After analyzing the reasons why distributed development fails, we decided to focus on communications, openness and transparency. So instead of face-to-face meetings, we turned to online solutions for solving a two-fold problem: integrating the parallel work of all developers nicely, and – more importantly – getting the people to form a self-organizing team.

We assembled an array of free/libre software tools around one core application, Trac, which integrates a wiki, version control, and ticket system into one whole, and is easy to customize and expand. The major challenge has been to get this heterogenous collection to actually supports development. Technical integration isn't enough, since many issues need social contracts, rules of conduct.

In this session we will go through the lessons we've learned during our project – what has worked, and what has failed. We also show how Trac needs to be customized and how its weak spots should be handled.

If you're involved in (or planning) a geographically distributed development team, you should definitely attend this session, both to get ideas on how to accomplish it, and to share your ideas. We also show the popular Trac system in action and show how it needs to be customized and what social rules need to be in place for it to support development properly. This session also presents a case study of applying XP and Scrum in a project that has high priority on usability and design.

3.7.2 Biography

Tarmo Toikkanen (PsM) is an entrepeneur, software engineer, researcher, and instructor. He's worked as a software architect/designer/programmer for 14 years, using technologies such as C/C++, Java, Python, Perl, PHP, (whatever)SQL, J2EE, and Zope. He's worked as an ICT instructor for 10 years, as a freelance journalist for 12 years, and is a doctoral student of educational psychology. Tarmo works as scrum master / architect / developer of the project being described in this tutorial.

Teemu Leinonen holds over a decade of experience in the field of research and development of web-based learning. His areas of interest and expertise covers design for learning, computer supported collaborative learning (CSCL), online cooperation, learning software design, educational planning and educational politics. Since 1998 Teemu has led the Learning Environments research group of the Media Lab, University of Art and Design Helsinki. Teemu conducts research and publishes in different forums. He has delivered a number of speeches in national and international conferences, has given in-service courses for teachers and has carried out consulting and concept design for several Finnish ICT and media companies.

4. Keynote Speeches

4.1 The Hacker Ethic: What Drives Human Action at Its Best?

Pekka Himanen
Professor at the University of Art and Design Helsinki, Finland
Visiting professor at University of Oxford, United Kingdom
Principal scientist at the Helsinki Institute for Information Technology, Finland

4.1.1 Introduction

What is the creative work ethic that characterizes the most achieving individuals and communities? Pekka Himanen sums up the challenge with the concept of 'hacker ethic', on which he wrote a book together with Linus Torvalds. Here, a hacker does not refer to computer criminals but to what the word originally meant: people who have a creative passion to what they do and want to do it together with others.

4.1.2 Biography

Pekka Himanen is one of the internationally best-known researchers of the information age, whose works on the subject have been published in 20 languages from Asia to America (English, Chinese, Japanese, Korean, Taiwanese, Indonesian, Russian, Ukrainian, Turkish, Portuguese, Spanish, Catalan, French, Italian, German, Dutch, Croatian, Estonian, Swedish and Finnish).

After obtaining his PhD in Philosophy as the youngest doctor ever in Finland at the age of 20 (University of Helsinki 1994), Himanen moved to carry out research first in England and then in California (Stanford University and the University of California, Berkeley). The best-known publication of this research is the book The Hacker Ethic (Random House 2001). Himanen has also coauthored with Prof. Manuel Castells the influential book The Information Society and the Welfare State (Oxford University Press 2002), which has been discussed worldwide in the leading academic and political circles. Himanen is nowadays a popular lecturer around the world.

As a sign of his impact, Himanen's work has been recognized with several awards, such as the World Economic Forum's respected Global Leader for Tomorrow Award in 2003. Dr. Himanen has also had an important role in the actual making of the information society policy. Recently, he has advised the International Labor Organization's High Level Commission on Globalization on the building of a socially sustainable global information society. In Finland, Dr. Himanen has recently finished the preparation of a new information society strategy for the Finnish Parliament's Committee for the Future.

Currently, Himanen divides his time between being a professor at the University of Art and Design Helsinki, a visiting professor at Oxford, and a principal scientist at the Helsinki Institute for Information Technology (a joint research center of the Helsinki University of Technology and Helsinki University). To contact him, email pekka.himanen@hiit.fi

4.2 Product and Process Architectures for Integrating Agile and Plan-Driven Methods

Barry Boehm
TRW Professor of Software Engineering
Computer Science Department Director
USC Center for Software Engineering, U.S.A.

4.2.1 Introduction

This talk will summarize the experiences of the USC Center for Software Engineering's industry affiliates in using XP and other agile methods. They find that the agile methods are effective on small projects, but difficult to scale up. The talk will then present a set of product and process architecture frameworks that we and the industry affiliates are finding effective for doing risk-driven integration of agile and plan-driven methods to fit the particular large-project elements' needs to simultaneously accommodate rapid change and high assurance.

4.2.2 Biography

Barry Boehm received his B.A. degree from Harvard in 1957, and his M.S. and Ph.D. degrees from UCLA in 1961 and 1964, all in Mathematics. Between 1989 and 1992, he served within the U.S. Department of Defense (DoD) as Director of the DARPA Information Science and Technology Office, and as Director of the DDR&E Software and Computer Technology Office. He worked at TRW from 1973 to 1989, culminating as Chief Scientist of the Defense Systems Group, and at the Rand Corporation from 1959 to 1973, culminating as Head of the Information Sciences Department. His current research interests include software process modeling, software requirements engineering, software architectures, software metrics and cost models, software engineering environments, and knowledge-based software engineering. His contributions to the field include the Constructive Cost Model (COCOMO), the Spiral Model of the software process, the Theory W (win-win) approach to software management and requirements determination and two advanced software engineering environments: the TRW Software Productivity System and Quantum Leap Environment. He has served on the board of several scientific journals, including the IEEE Transactions on Software Engineering, IEEE Computer, IEEE Software, ACM Computing Reviews, Automated Software Engineering, Software Process, and Information and Software Technology.

4.3 Responsible Development: Making the Most Out of Reality

Kent Beck
Founder and Director of Three Rivers Institute (TRI), U.S.A.

4.3.1 Introduction

What does it mean to develop responsibly in 2006? What are we doing right and why? What are the implications of this style of development?

4.3.2 Biography

Kent Beck is the founder and director of Three Rivers Institute and an Agitar Software Fellow. His career has combined the practice of software development with reflection, innovation, and communication. His contributions to software development include patterns for software, the rediscovery of test-first programming, the xUnit family of developer testing tools, and Extreme Programming. He currently divides his time between writing, programming, and coaching. Beck is the author/co-author of Extreme Programming Explained: Embrace Change 2nd Edition, Contributing to Eclipse, Test-Driven Development: By Example, Planning Extreme Programming, The Smalltalk Best Practice Patterns, and the JUnit Pocket Guide and the forthcoming Implementation Patterns. He received his B.S. and M.S. in Computer Science from the University of Oregon.

4.4 Mad Software Disease

Sean Hanly
Exoftware, Ireland

4.4.1 Introduction

Over the last several years Sean Hanly has worked with numerous companies helping them to transition to Agile delivery. During that time he has discovered a new, and as yet to be recognised disease running rampant throughout software delivery organisations – Mad Software Disease (MSD). At its simplest, organisations with this disease insist on taking the same approach over and over again, but expect a different result – clearly delusional behaviour. You might already know your organisation suffers from MSD – but how do you eradicate such a deadly disease? In this talk Sean will help you to diagnose your organisation for MSD and analyse symptoms and indications. He will give you new and unique ways to communicate these symptoms and their causes to others within your organisation. He will also look at how taking an Agile approach is the cure to get your organisation back on the road to health!

4.4.2 Biography

Sean Hanly is a co-founder and director of Exoftware, an Agile mentoring and development company. Sean currently heads Exoftware's delivery services overseeing Agile transitions, as well as working hands on as a mentor and trainer for clients throughout Europe. As well as client-based work, Sean has lectured on Agile methods at University College Dublin and worked closely with the DSDM Consortium on the release of XP and DSDM. Sean also spends time speaking at industry events, and writing on Agile topics for various industry publications such as Computer Weekly, IT Week and Application Development Trends. He has been instrumental in kick starting Exactor, an OpenSource acceptance testing framework, currently being used by several Agile teams. Prior to founding Exoftware, Sean held programming, technical architect and team lead positions with companies such as Allied Irish Bank, Jeffries and Company and American Management Systems, where he was awarded a Principal position – one the youngest people ever to achieve the position at AMS. Sean received his honours Bachelor of Commerce and honours Masters in Information Technology degrees from University College Galway. Sean is also currently the CTO and board member of 3Q Solutions Limited.

4.5 Extreme and Lean in LARGE Projects

Jack Järkvik, Vice-President, Ericsson AB, Sweden
in conversation with Erik Lundh

4.5.1 Introduction

Conventional wisdom tells us that XP work only in small teams and projects. At Ericsson, the very first GSM Radio Base Station project, a ca 1990 multi-site project with hundreds of people involved, was saved by switching to a very XP-like approach. The new approach changed not only the interface but the core flows of the whole project. GSM systems have been a growth business for Ericsson since. The approach has been used ever since in select large critical projects within Ericsson and with many of their subcontractors.

The key to large scale extreme development seems to be neither patterns nor values – success is dependent on cross-competency experiences of truly successful agile projects with lots of creativity and momentum, and still with minimum waste.

Not everyone might have opportunity to first hand experience the creative hands-on processes of world class carmakers like Toyota (or in Jacks case, growing up with the designers that made SAAB a premium brand).

Is there hope for the rest of us to be successful in large agile projects with hundreds of people? Yes, if we give people a chance to gain first hand experience in small but hot agile projects where both soft and technical aspects help bring home the success.

Extreme Programming, XP, is one excellent candidate to bring that "what matters in projects" experience to both managers and developers, laying the necessary foundation of understanding for large scale agility.

Join seasoned XP-coach Erik Lundh in a conversation with Jack Järkvik on how we can gain the right experiences that help us make big projects Extreme and Lean.

You will hear about some of the practices that make select large projects Extreme and Lean at Ericsson. And you will get opportunity to ask why we recommend XP over SCRUM to gain experience for large scale agile projects. Motto: Build insight, not pyramids!

4.5.2 Biography

Jack Järkvik is educated at Gotheburg University, Chalmers University of Technology and Sloan School of Management, MIT. Nearly thirty years in telecom of which driving his own consulting firm one decade. Järkvik is now reemployed at Ericsson, presently a vice president of R&D Operational Excellence. He has dealt with system development nearly all his career ranging from AXE, GSM as well as CDMA and WCDMA.

Appendix 1: Author Index

Published by	Series title, number and report code of publication VTT Symposium 241 VTT–SYMP–241

Author(s)
Salo, Outi, Abrahamsson, Pekka & Jaring, Päivi (eds.)

Title

The 7th International Conference on eXtreme Programming and Agile Processes in Software Engineering
Tutorials, Workshops, Activities, and Keynote Speeches

Abstract
This proceedings is a collection of all the tutorials, workshops, activities and keynote speeches of the 7th International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP 2006) held in June 17–22, 2006, Oulu, Finland.

Keywords
software engineering, eXtreme programming, agile processes, adaptive processes, agile development, software development, XP projects, testing

ISBN
951–38–6305–0 (soft back ed.)
951–38–6306–9 (URL: http://www.vtt.fi/publications/index.jsp)

Series title and ISSN VTT Symposium 0357–9387 (soft back ed.) 1455–0873 (URL: http://www.vtt.fi/publications/index.jsp)			Project number 549
Date June 2006	Language English	Pages 92 p. + app. 3 p.	Price B
Name of project XP2006		Commissioned by	
Contact VTT Technical Research Centre of Finland Kaitoväylä 1, P.O. Box 1100 FI-90571 OULU, Finland Phone internat. +358 20 722 111 Fax +358 20 722 2320		Sold by VTT Technical Research Centre of Finland P.O.Box 1000, FI-02044 VTT, Finland Phone internat. +358 20 722 4404 Fax +358 20 722 4374	

VTT SYMPOSIUM 241 The 7th International Conference on eXtreme Programming and Agile...

This proceedings is a collection of all the tutorials, workshops, activities and keynote speeches of the 7th International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP 2006) held in June 17–22, 2006, Oulu, Finland.

Tätä julkaisua myy	Denna publikation säljs av	This publication is available from
VTT	VTT	VTT
PL 1000	PB 1000	P.O. Box 1000
02044 VTT	02044 VTT	FI-02044 VTT, Finland
Puh. 020 722 4404	Tel. 020 722 4404	Phone internat. +358 20 722 4404
Faksi 020 722 4374	Fax 020 722 4374	Fax +358 20 722 4374

ISBN 951–38–6305–0 (soft back ed.)
ISSN 0357–9387 (soft back ed.)

ISBN 951–38–6306–9 (URL: http://www.vtt.fi/inf/pdf/)
ISSN 1455–0873 (URL: http://www.vtt.fi/inf/pdf/)