



PEALING THE HYPE INTO PIECES: WHAT DO WE REALLY KNOW
ABOUT AGILE IN RESEARCH AND PRACTICE?

Pekka Abrahamsson

VTT TECHNICAL RESEARCH CENTRE OF FINLAND

Oliopäivät, 29.11.2006



AGENDA TODAY

- Motivation
- Scientific results
- Practical cases
 - Engisud, Ita
 - Nokia, Fin



ON SCIENTIFIC STUDIES

- Why should we care about scientific results?



- Slow: Takes long time to make to a conference (6-12 months), takes even longer for a good journal (1-3yrs)
- Uncontrollable: Scientists are opportunistic people
- Practicality: Very difficult to generalize, experiments often performed in a Mickey-Mouse environment
- Goal: Nothing is as practical as a good theory
- Practicality: rules-of-thumb, patterns, laws, understanding
- Validity: peer review process

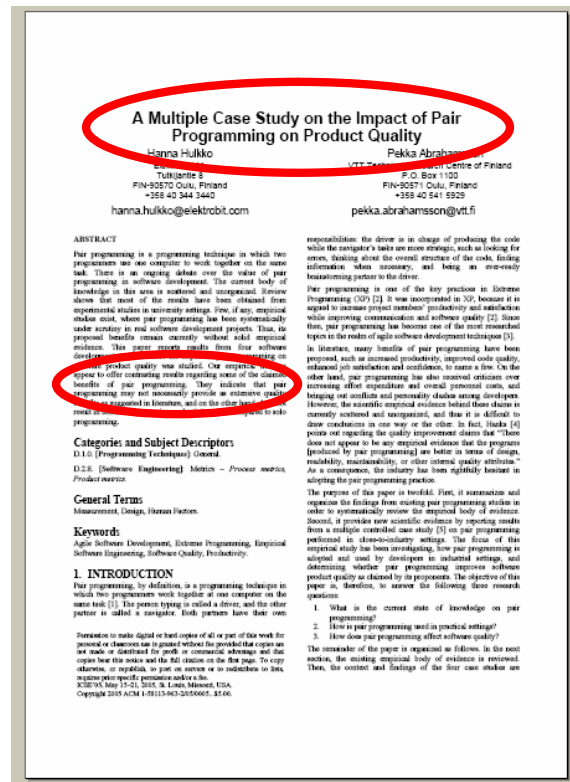


ON READING SCIENTIFIC STUDIES

- Not all scientific studies are equal
- Not all written in paper is true or objective
- A good paper has only one or two points it tries to prove
- A scientific study is not a detective story. there is no suspense saved to the end, everything is revealed in title, abstract and introduction.
- If you do not understand the paper, title objective, or results, the authors failed, not you.

ON READING SCIENTIFIC STUDIES

A Multiple Case Study on the Impact of Pair Programming on Product Quality

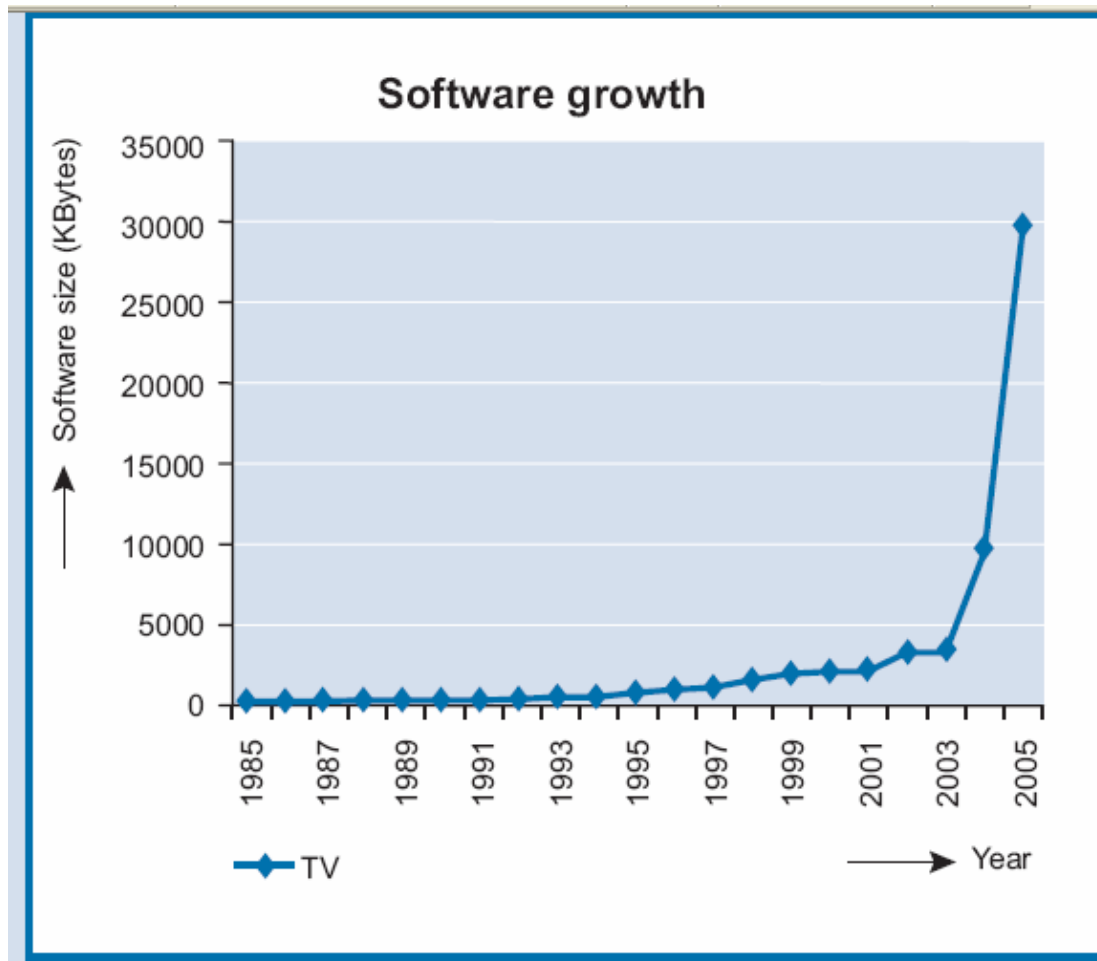


They indicate that pair programming may not necessarily provide as extensive quality benefits as suggested in literature, and on the other hand, does not result in consistently superior productivity when compared to solo programming.

A GOOD PAPER HAS ONLY ONE OR TWO POINTS. AND THEY ARE TOLD ALREADY IN THE ABSTRACT



The amount of software in embedded devices grows faster today than the Moore's Law (Philips)



30.11.2006



INDUSTRY ACCEPTANCE

Home > Topics > IT Management > News > **Microsoft Lauds 'Scrum' Method for**

Microsoft Lauds 'Scrum' Method for Software Projects

By Darryl K. Taft

November 11, 2005



▶ Be the first to comment on this article

When Microsoft launched its long-awaited database and tools products last week, the company acknowledged it would have to act faster to revise its products faster as customer needs grow.

One way Microsoft's development teams intend to deliver on this is through the use of agile development methodologies, such as extreme programming and Scrum, company officials said.

Scrum is an agile method for management of software development projects.

David Treadwell, corporate vice president of the .Net Developer Platform group at Microsoft, said that while Microsoft welcomes the use of methodologies like Scrum, "we're not mandating them, but we're encouraging them. So Scrum is one process—the idea that teams meet once a day for half an hour, figure out what they're

Source:
eweek.com



» SAP INFO international: Home » Strategy

Article	Author
---------	--------

July 26, 2006 II SAP sees promise in Agile Software Development:

Customer Involvement at All Stages

Agile Software Development (ASD) is today the most talked about methodology for quickly creating code and many large companies, including SAP, are either piloting or have fully embraced this relatively new way of creating software.

Since the dawning of the computer age, a cause célèbre has been to improve the efficiency of programming to create better software products, more quickly. Go back a few years and the phrases and acronyms such as object oriented, 3GL and 4GL – third and fourth generation language – were on many people's lips.

Within ASD there are several complementary technologies. They include Extreme Programming, Adaptive Software Development, Lean Software Development and Scrum.

What all the ASD processes have in common, according the Agile Alliance, a non-profit organization which was started in 2001 by several industry luminaries to propagate the techniques, is "emphasized close collaboration between the programmer team and business experts; face-to-face communication (as more efficient than written documentation); frequent delivery of new deployable business value; tight, self-organizing teams; and ways to craft the code and the team such that the inevitable requirements churn was not a crisis."

Development on the fly

In a nutshell that means that the traditional ways of developing software are out of the window. In traditional "plan driven" development developers and customers agree on the specification at the front-end of a project. The developers then go away and in a few months time deliver the product.

Source:
www.sap.info





Welcome to ASCI

Agile Software Community of India (ASCI) is a registered society founded by a group of agile enthusiasts and practitioners from companies that practice Agile Software Development methodologies.

ASCI is formed to create a platform for people from different software organizations to come together and share their experience with Software development methodologies. ASCI's focus is Agile and related light weight methodologies/philosophies. ASCI evangelizes itself to be a facilitating body which fosters and innovates lightweight methodologies in software development in India. ASCI is working with Universities and Student chapters to increase Agile awareness within the academic circles.

So far ASCI have successfully organized five conferences on Agile in India. For more details on these conferences visit the [Events](#) page. These conferences are initiatives to reach out to other agile practitioners and to increase the awareness of Agile Methodologies in India.

We intend to have periodic meetings and workshops to share experiences from the field and improve our practice. If you'd like to join ASCI in an individual or corporate capacity, please visit the [Membership](#) page.

Never heard of Agile? Browse through this [road map](#) from [Agile Alliance](#) to get a better idea.

News

[Sanjiv Augustine to kick off Chennai Agile User Group Meeting on July 8th 2006](#)

[Agile India 2006 presentations now available for download](#)

[Online registrations for Agile India 2006 are closed. On-the-spot registration subject to availability.](#)

Agile development considered

- Agile is not an absolute concept
- There is no definite set of agile practices
- Agile manifesto is the first of its kind in the field
- Agile concept originated from the practitioner/consultant perspective
- Agile solutions contribute to a natural concern of software developers – i.e., e.g., concern for efficiency
 - > Agile communities established, developers participate & contribute to conferences, etc.

**Fact corner:
Developers have accepted
agile methods well**

- Nokia's slides removed. The can be downloaded freely from

http://www.odd-e.com/articles/2006/nokia_agile.pdf

EMPIRICAL BODY OF EVIDENCE

- Agile methods have been criticized for the lack of sound empirical data to validate their claims: e.g.,
 - *“Presently, there is a lot of anecdotal evidence and **very little empirical and validated data of agile methods effectiveness**. Much of the evidence is based on the stories and preferences of those who practice it.”* [Melnik et al. 2002]
 - *“In recent years, there have been many stories and anecdotes of industrial teams experiencing success with Agile methodologies. There is, however, **an urgent need to empirically assess the applicability of these methods...**”* [Lindvall et al. 2002]
 - *“**Empirical evidence, based on rigorous research, is scarce**. Only three out of nine [agile] methods have some empirical support for their claims.”* [Abrahamsson et al. 2003]



AGILE METHODS & BUSINESS IMPACT?

Google Scholar BETA

agile methods business impact Search

Advanced Scholar Search
Scholar Preferences
Scholar Help

Scholar All articles Recent articles Results 1 - 10 of about 11,400 for agile methods business impact

All Results

[P Abrahamsson](#)
[J Warsta](#)
[J Ronkainen](#)
[R Baskerville](#)
[M Fowler](#)

[New directions on agile methods: A comparative analysis - group of 8 »](#)
P Abrahamsson, J Warsta, MT Siponen, J Ronkainen - International Conference on Software Engineering (ICSE25), ..., 2003 - doi.ieeecomputersociety.org
... them fast, collecting feedback and reacting rapidly to **business** and technology changes [7-9]. The emerging of numerous different **agile methods** has exploded ...
Cited by 49 - Related Articles - Web Search - Search Full Text at VTT - BL Direct

[Systems Without Method: The Impact of New Technologies on Information Systems Development Projects](#)
R Baskerville, J Travis, DP Truex - ... 2 Working Conference on The Impact of Computer Supported ..., 1992 - portal.acm.org
... Mark Lycett , Dilip Patel, **Business** modelling with ... Information systems security design **methods**: implications for ... New directions on **agile methods**: a comparative ...
Cited by 61 - Related Articles - Web Search

[Ten deadly risks in Internet and intranet software development - Search Full Text at VTT - group of 4 »](#)
D Reifer - Software, IEEE, 2002 - ieeexplore.ieee.org
... to traditional projects, the **impact** of unrealistic ... development with your **business** goals ... **methods**; object-oriented approaches, **Agile methods**; component-based ...
Cited by 23 - Related Articles - Web Search - BL Direct

[Agile software development methods - group of 12 »](#)
P Abrahamsson, O Salo, J Ronkainen, J Warsta - VTT Publications, 2002 - oasis.oulu.fi
... how the Open Source Software (OSS) paradigm places itself between the **agile** and plan-driven **methods**. The OSS is still fairly new in **business** environment and a ...
Cited by 95 - Related Articles - Web Search - Library Search - Search Full Text at VTT - BL Direct

[Empirical Evaluation of Agile Software Development: A Controlled Case Study Approach - group of 6 »](#)
O Salo, P Abrahamsson - Profes 2004, 2004 - Springer
... the highest scientific and **business impact** Target setting ... variables Multiple data collection **methods**: Qualitative and ... literature review of **agile/XP methods** [45 ...
Cited by 9 - Related Articles - Web Search - BL Direct

[\[book\] Software Engineering: A Practitioner's Approach - group of 3 »](#)
RS Pressman - 2004 - books.google.com
... focuses on software and its **impact** on **business** and society. ... Based Development 59 3.5.2 The Formal **Methods** Model 60 ... CHAPTER 4 **AGILE** DEVELOPMENT 7 1 4. 1 What Is ...
Cited by 2617 - Related Articles - Web Search - Library Search

[Breaking the ice for agile development of embedded software: an industry experience report - group of 3 »](#)
P Manhart, K Schneider, DCR Center, G Ulm - Software Engineering, 2004. ICSE 2004. Proceedings. 26th ..., 2004 - ieeexplore.ieee.org
... the production plan has a major **impact** on the ... In our case, the **business** unit saw little ... **Agile methods** have provoked a controversy between "classical" and ...
Cited by 10 - Related Articles - Web Search - Search Full Text at VTT - BL Direct

11400 hits!

3 hits
In the top ten list
are to my own work?

SITUATION IS FAR FROM UNIQUE

- This is not, however, new realization in the field of SW Engineering: e.g.,
 - While software professionals seek rational basis for making a decision which software engineering method they should adopt, the basis for such a rationalization is completely missing. Methods introduced continue be based more on faith than on an empirical data. [Fenton 2001]
 - *“Experimentation is central to the scientific process. [...] Without experiments, computer science is in danger of drying up and becoming an auxiliary discipline. The current pressure to concentrate on application is the writing on the wall”* [Tichy 1998]

THE CREDIBILITY OF AGILE MOVEMENT IS AT STAKE?

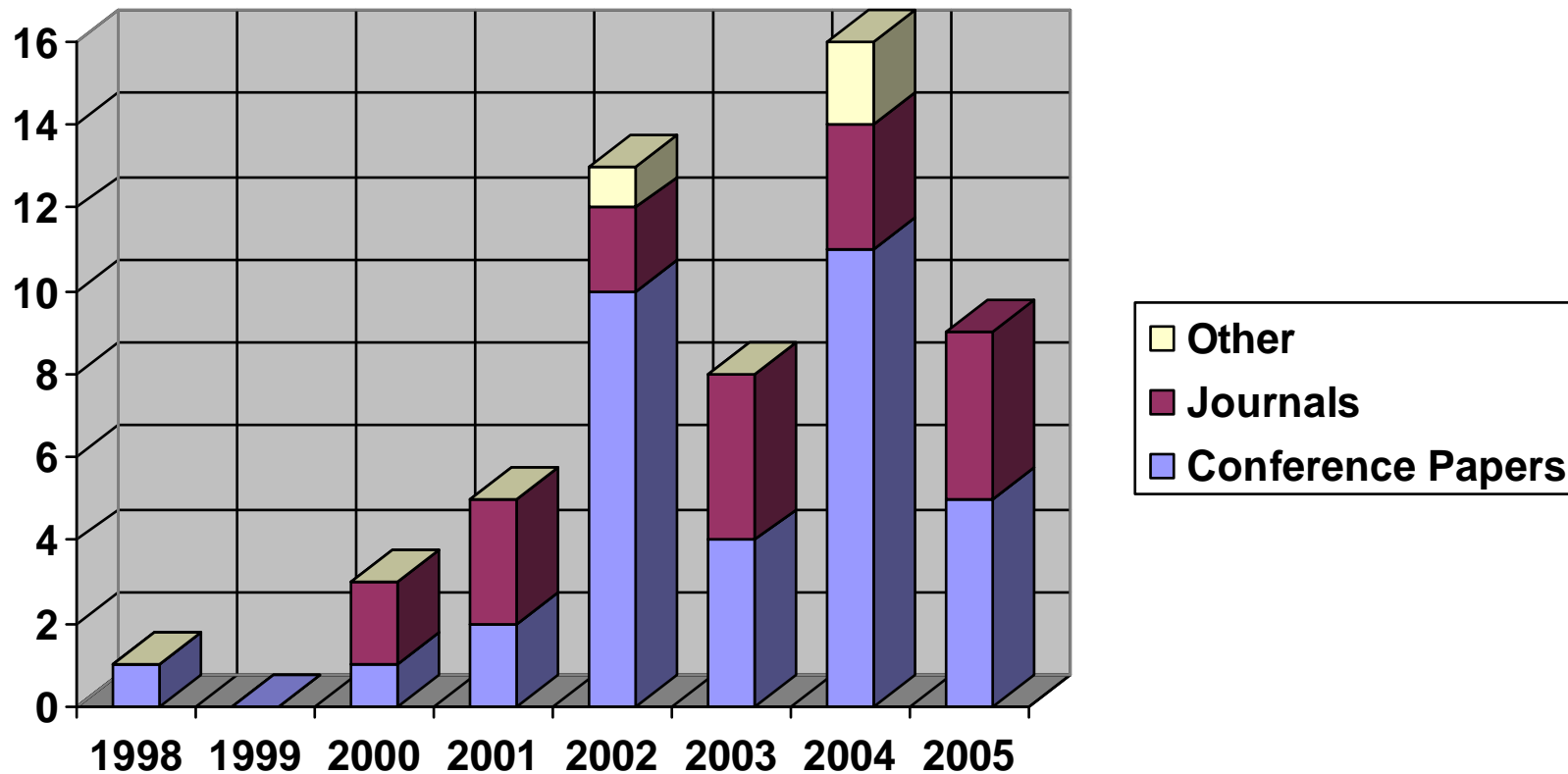
- To sum it up. The fundamental basis for agile movement can not rely on leading professionals' assertions based on their personal view:
 - "I haven't ever built missile nosecone software, so I don't know what it is like."
[Beck 2000]
 - Based on the above, can you recommend someone developing safety-critical software to even think about agile methods?



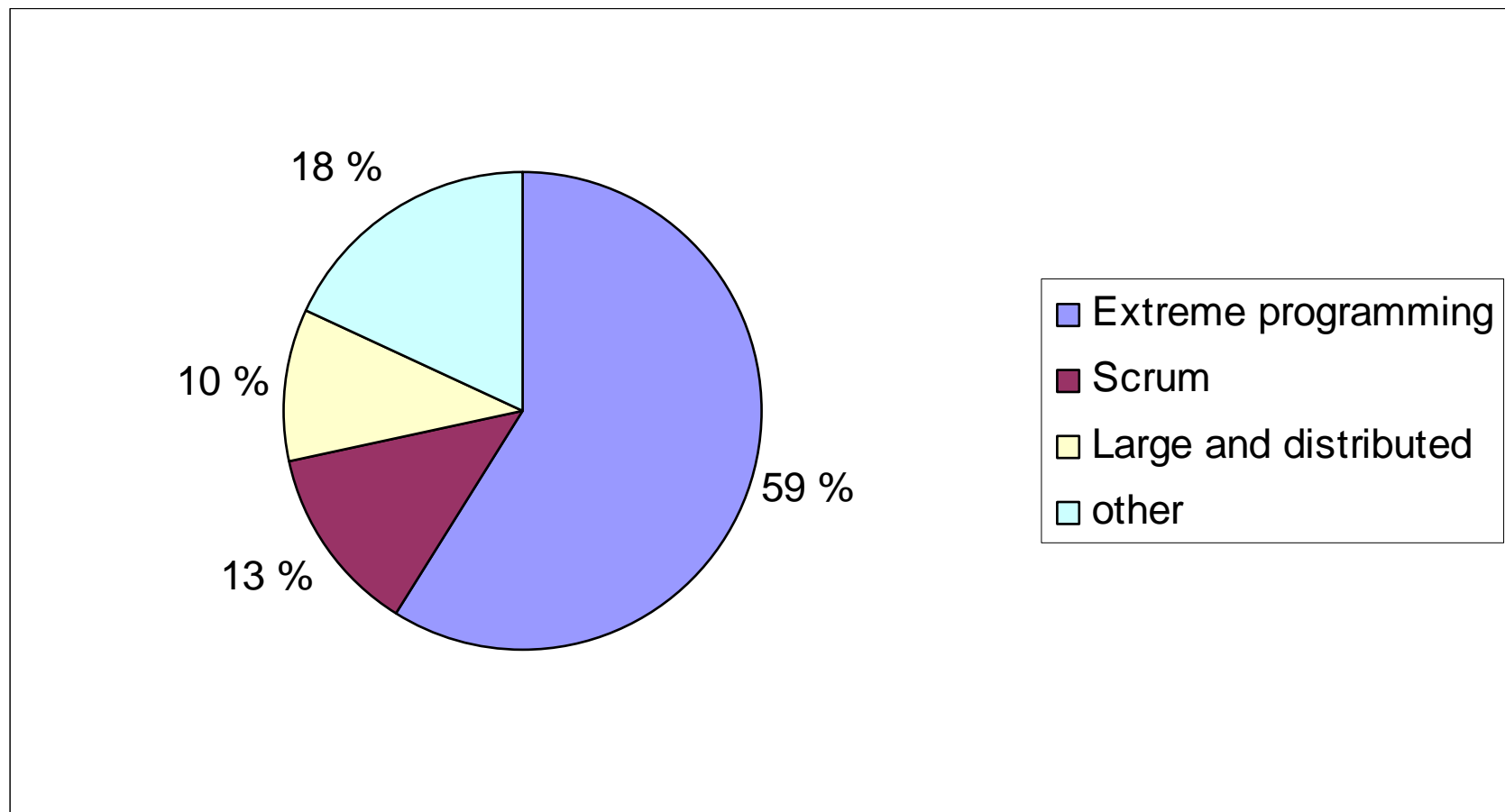
EMPIRICAL STUDIES ON AGILE METHODS

- A database search into ACM, IEEE digital libraries, Springer-verlag conference publications, XP 2000-2005, google-search, was performed
- Results revealed 55 papers, 36 based empirical evidence
 - 4 surveys
 - 9 experiments
 - 9 case studies
 - 14 experience reports

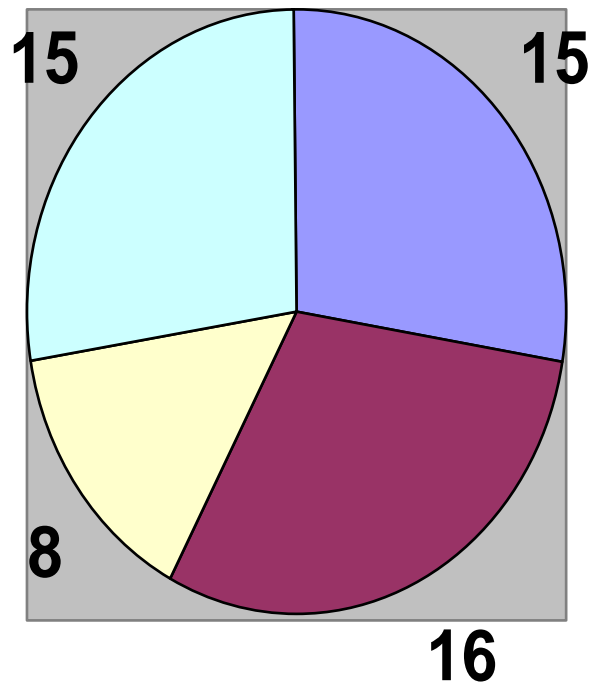
CURRENT STATUS OF EMPIRICAL AGILE PAPERS



FOCUS OF AGILE STUDIES



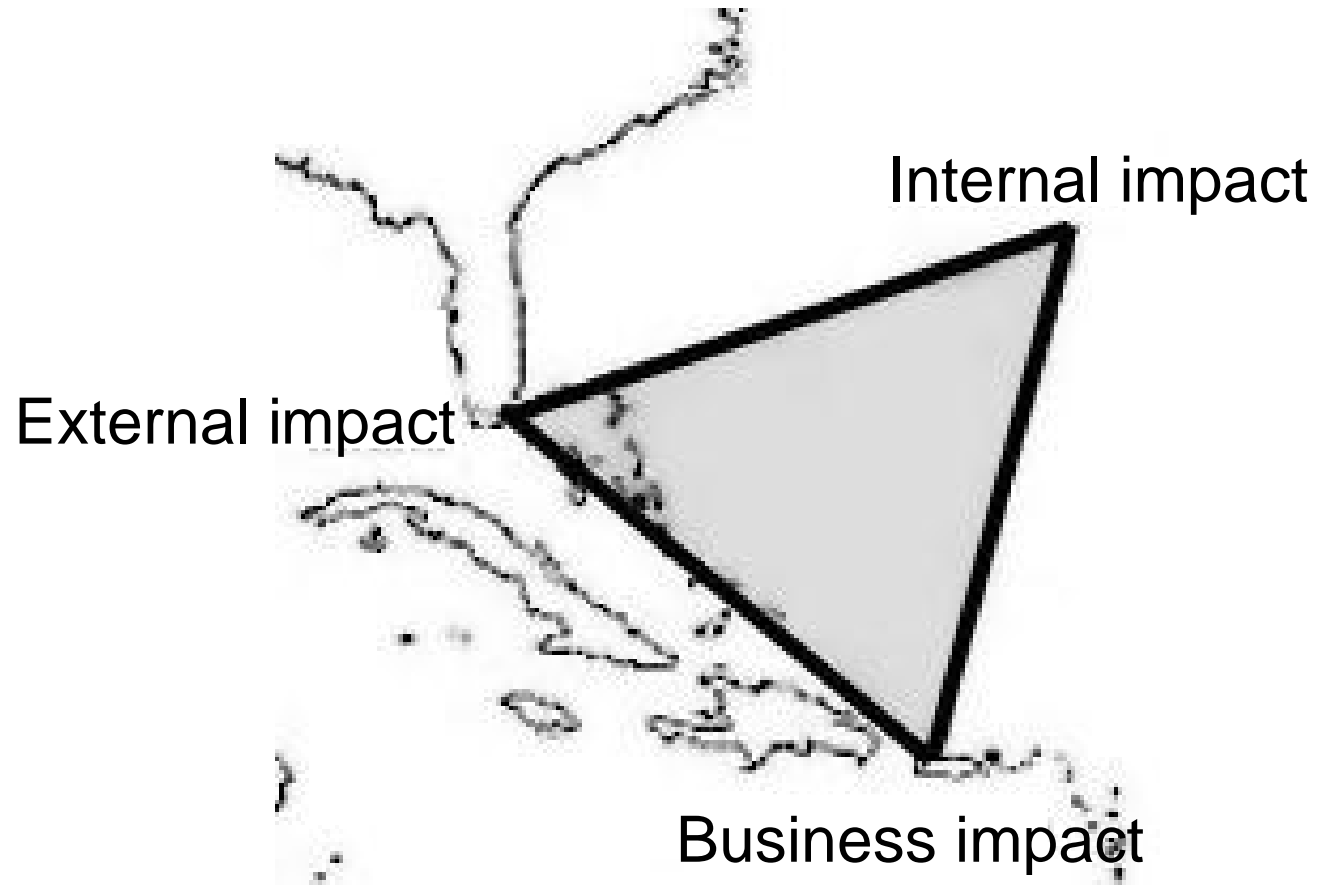
CONTEXT OF AGILE STUDIES



- Small Industrial Context
- Medium Industrial Context
- Large Industrial Context
- Student/Academic Environment

Fact corner:
Small < 10 persons
Medium < 50 persons
Large > 50 persons

ORGANIZATION OF THE REVIEW RESULTS



POSITIVE IMPACT IN INDUSTRY

Positive Experience	Rationalization	Impact
Improved communication and feedback	Achieved via short iterations, pair programming and open office space	Internal
Project status	Daily Stand Up Meetings	Internal
Quality of the code	Test Driven Development, Short Iterations	Internal
Requirements can change	Planning, Refactoring	Internal
Developer satisfaction	Pair Programming, XP in Overall	Internal
Customer satisfaction	Short cycles	External
Management satisfaction	Daily Stand Up Meetings, Time Tracking and Estimation	External
Better customer collaboration	Short Iterations	External

Improved communication and feedback

- Achieved via short iterations
 - Cycles have provided closer feedback (Lippert, Becker-Pechau et al. 2003)
 - Scrum radically improves the communication i.e. daily stand up meetings (Sutherland 2001)
 - Higher quality by earlier feedback from the customers (Aoyama 1998)
- Pair programming
 - Has a significant, positive influence on the satisfaction of developers (due to increased communication, speed of communication of design changes, and organizations of meetings) (Succi et al. 2002)
- Open office space
 - Extroverted people benefit working in an open office space (Law and Charron 2005)

INTERNAL IMPACTS

- **Better customer collaboration** achieved due to shorter Iterations
 - A trust boost between customers and inside the company due to incremental releases (Howard 2003)
- **Project status** awareness significantly improved due to daily stand up meetings
 - stand-up meetings kept everyone aware of the project status (Rasmusson 2003)
- **Quality of the code**
 - TDD - Developers produce higher quality code than traditional, water-fall like developers (George and Williams 2003)
 - Early feedback - Higher quality by earlier feedback from the customers (Aoyama 1998)

CHANGING REQUIREMENTS PERCEIVED POSITIVELY

- Planning
 - Aoyama (1998)
 - higher stability of work-loads
 - higher utilization of work-loads
 - higher flexibility to change development plans
 - Incremental planning, evolutionary design and efficient spike can reduce the scope of change. Thus, rework scope is reduced with a small cost (Bin, Xiaohu et al. 2004)
- Refactoring
 - Design flaws can be found (Morsicato and Poppendieck 2002)

Developer satisfaction

- Pair Programming
 - Pair Programming socially rewarding due to its nature to raise morale (Gittins, Bass et al. 2004)
 - Pair programming has a significant and positive influence on the satisfaction of developers due to increased communication, speed of communication of design changes, and organizations of meetings (Succi et al. 2002)
- XP in overall
 - Morale among team 11% increased; Morale in the scale of 0 -10: 6.8 (Williams 2004)

DEVELOPER SATISFACTION

Source: Nokia, publicly available at:
http://www.odd-e.com/articles/2006/nokia_agile.pdf

Customer satisfaction

- High customer satisfaction: qualitative and anecdotal evidence (Williams 2004, Mann and Maurer 2005)
 - Short cycles

Management satisfaction

- Daily stand up meetings
 - In one case, Scrum eliminated several hours a day of senior management meeting time starting the day Scrum began (Sutherland 2001)
- Time tracking and estimation
 - (Cohn and Ford 2003)
 - An organization with several incorrent project estimation: helps to convince management to try something new approaches
 - Model status reports can help convince upper management that agile processes allow adequate project tracking

Negative Experiences in Industry

Negative Experience	Rationalization	Impact
Introducing agile software development (especially into large organizations)	Tailoring practices Test-Driven Development Small releases Organizational and cultural reasons Conflicts between traditional and agile teams	Internal
Need for better tools	Test-Driven Development Continuous integration Refactoring	Internal
Pair Programming	Skills of the programmers Nature of the programmers	Internal
Resistance	Agile Principles Management Resistance	Internal and External
Customer commitment	Meeting, Planning, Testing, Schedule	External

Resistance

- Agile Principles
 - Programmers preferred to implement things the old way: producing noncode artifacts, such as design documents. Programmers preferred plan-driven methods saying anything else is too chaotic: they believed plan-driven processes looked better on a resume (Cohn and Ford 2003)
 - E.g. refactoring can be seen as process failure in traditional development i.e. changing the code frequently. (Lindvall, Muthig et al., 2004)
- Management resistance
 - Although the management had strong refusal at first, eventually they accepted XP due to adequate experience on trial projects (Grenning 2001)

Customer Commitment

- Meeting
 - Risk management gets complicated if the customer cannot attend releases: the team does not know what to actually implement (Lippert, Becker-Pechau et al., 2003; Rumpe and Schröder 2002)
- Planning
 - Story planning may get complicated if there is no business customer present (Rasmusson 2003)
- Testing
 - Customers get little benefit of testing in the early phases, so they may not test and evaluate prototypes as promised (Lippert, Becker-Pechau et al, 2003)
- Schedule
 - External services and expertise were not able to be in-line with the two week-iterations: assumption that they may caused a conflict (Rasmusson 2003)

Introducing Agile Software Development (Especially into Large Organizations) 1/2

- Tailoring practices
 - Without tailoring, introducing agile in large organizations is infeasible, because e.g. XP is not a one-size fits-all software development process (Lindvall, Muthig et al., 2004)
- Test-Driven Development
 - At first, hard to adopt and to use (Karlström 2002; Schuh 2001) due to variety of issues in the target environment (Svensson and Höst 2005)
- Small releases
 - Reported as hard to introduce (Karlström 2002)

Introducing Agile Software Development (Especially into Large Organizations) 2/2

- Organizational and cultural reasons
 - Individual projects in a large organization often depend on their environment in several ways (Lindvall, Muthig et al., 2004; Svensson and Höst 2005)
 - Lack of skills, such as programming, testing and design decreased the level of communication (Howard 2003)
- Conflicts between traditional and agile teams
 - Usually agile and traditional teams produce conflicting products (Boehm and Turner (2005)
 - XP teams had difficulties interfacing with teams employing traditional approaches (Drobka, Noftz et al., 2004; Lindvall, Muthig et al., 2004)
 - Organizational software processs has its own requirements management. XP team had do to twice amount of work due to traditional requirements management (Lindvall, Muthig et al., 2004)

Need for Better Tools

- Test-Driven Development
 - Reported that better tools would have been needed without particular reasons (Howard 2003)
- Continuous integration
 - Continuous integration caused Configuration Control Board (CCB) to clash (Lindvall, Muthig et al., 2004)
- Refactoring
 - Reported that better tools would have been needed without particular reasons (Howard 2003)

Pair Programming

- Skills of the programmers
 - Pairing members with different skills may cause the unskilled one to not take part in programming (Grosman, Bergin et al., 2004)
 - Pair Programming pairs should be chosen carefully: two inexperienced is not good because it may take too much time for them to learn how to solve complicated tasks (Drobka, Noftz et al. 2004; Gittins, Bass et al. 2004)
- Nature of the programmers
 - Some of the experienced programmers felt that their status was lowered when joining an XP team (Gittins, Bass et al., 2004)
 - Pair Programming pairs should be chosen carefully: shy + extrovert cause the shy one to not participate (Drobka, Noftz et al. 2004)

Business impacts

- Release time reduction?
- Overall quality?
- Test coverage?
- Productivity?
- The categories are used in different terms in various studies: some studies give exact percentual data but some of them rely only on experiences.

Release time reduction

- Small industrial context (Williams, Krebs et al., 2004, Williams 2004)
- Medium industrial context
 - 25-50% decreased, cost reduction 5-7% (Reifer 2002; Reifer 2002)
- Large industrial context
 - 75% (1 year --> 3 months) (Aoyama 1998)
- Amount of over time decreased (Mann and Maurer 2005)

Overall quality

- Small industrial context
 - ~45 % pre-release fewer defects (Williams, Krebs et al., 2004)
 - ~48% and 40% post-release fewer defects (Williams 2004)
 - improved quality 51% and 74% (Drobka, Noftz et al., 2004)
- Medium industrial context
 - Product quality improved (Reifer 2002; Reifer 2002)
- Large industrial context
 - No reported results on this review.

Test coverage

- Small industrial context
 - 16% improvement (Williams, Krebs et al., 2004)
- Medium industrial context
 - No reported results on this review.
- Large industrial context
 - No reported results on this review

Productivity 1/2

- Small industrial context
 - 70% improvement (Williams, Krebs et al., 2004)
 - 46% more codelines (Williams 2004)
 - 3,85-fold improvement compared to waterfall model (Drobka, Noftz et al., 2004)
- Medium industrial context
 - Improved (Svensson and Höst 2005)
 - functionality more 4 to 5 times industry average (Sutherland 2001)
 - increased 15-23% (Reifer 2002; Reifer 2002)

Productivity 2/2

- (Maurer and Martel 2002)
 - 66.3% increase in a number of produced locs/effort
 - 302.1% increase in a number of methods/effort
 - 282.6% increase in a number of classes/effort
 - 4.96% decrease in a number of features/effort
- Large industrial context
 - No reported results on this review

Summary of Impact

Release time reduction	20%-75% reduction
Overall quality	Pre-release: 40-65% reduction Post-release: 30-40% reduction
Test coverage	16% improvement
Productivity	from 15 to 70% improvement in productivity more functionality (4 to 5 times industry average) 46% more codelines

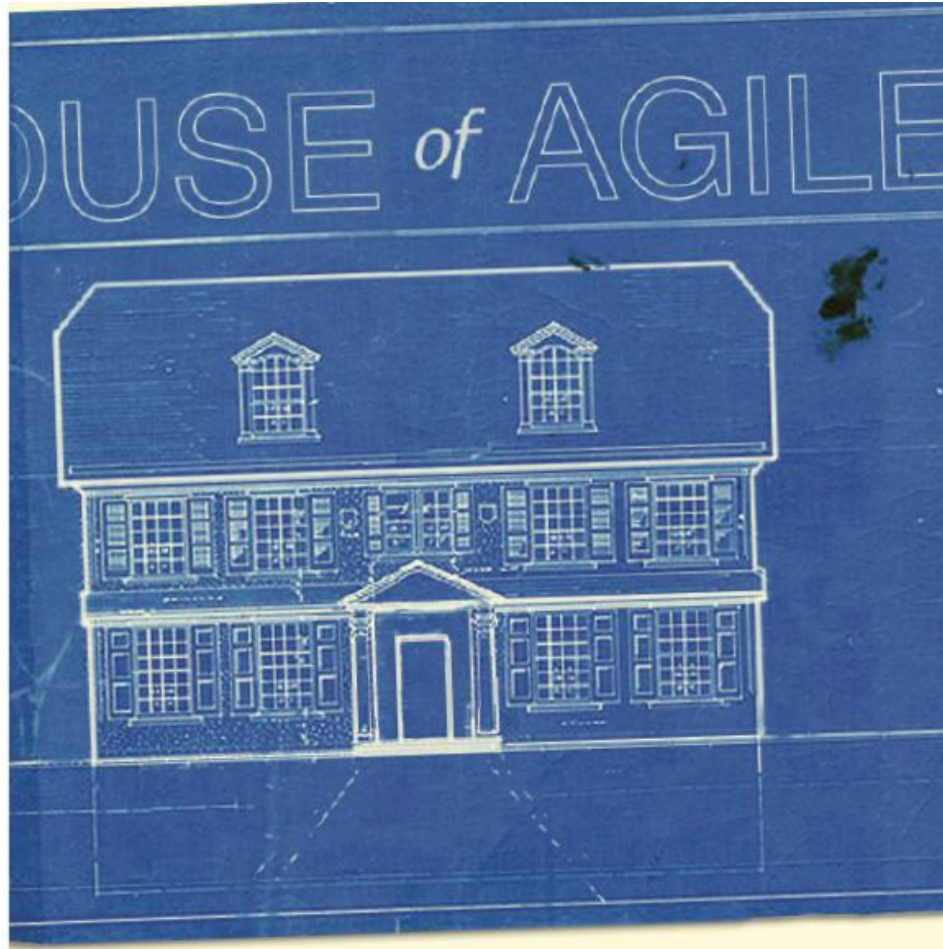
BASED ON THE CURRENT EMPIRICAL KNOWLEDGE, WHAT DO WE KNOW?

- Agile practices (TDD, pair programming, etc.) have predominantly delivered positive results
- Agile software development is scaling up
- Agile software development is applied in practically "all" development domains
- Higher {developer, management, customer} satisfaction rates achieved generally
- Communication within the team and outside tends to improve dramatically

CONCLUSIONS

- The body of empirical evidence in applying agile methods is growing steadily
- Unlike many other software engineering innovations, some impact figures are already available
 - However, only few failures are reported
- Agile methods keep spreading quite rapidly in new domains (e.g., Avionics)
- Standardization work ongoing
 - IEEE 1648 released by the end of 2007

HOUSE OF AGILE - AGILE SOLUTION FRAMEWORK



- 80 Industrial pilots
- Scientific publications
- 1 book in the publication process
- Processes
 - New methods, production frameworks
- Training material
 - Presentations, courses, Videos

<http://www.houseofagile.org/>
Public release on January 30th, 2007

30.11.2006

REFERENCES

- Agile Alliance (2001). The Agile Alliance Web-Site. 2005.
- Ahmed, A., M. M. Fraz and F. A. Zahid (2003). Some results of experimentation with extreme programming paradigm. 7th International Multi Topic Conference, INMIC 2003, Islamabad, Pakistan.
- Aoyama, M. (1998). Agile Software Process and Its Experience. Proceedings of the 20th International Conference on Software Engineering, Kyoto, Japan.
- Aydin, M. N., F. Harmsen, K. Van Slooten and R. A. Stegwee (2004). "An Agile Information Systems Development Method in Use." Turkish Journal of electrical Engineering & Computer Sciences 12(2): 127-138.
- Baheti, P., E. Gehringer and D. Stotts (2002). Exploring the efficacy of distributed pair programming. XP/Agile Universe 2002, Chicago, Illinois, USA.
- Beck, K. (2000). eXtreme Programming Explained: Embrace Change. Reading, Massachusetts, Addison-Wesley.
- Beck, K., M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland and D. Thomas (2001). Manifesto for Agile Software Development. 2005.
- Benediktsson, O., D. Dalcher and H. Thorbergsson (2004). Working with Alternative Development Life Cycles: A Multiproject Experiment. 13th Conference on Information Systems Development, Vilnius, Lithuania.
- Bin, X., Y. Xiaohu, H. Zhijun and S. R. Maddineni (2004). Extreme programming in Reducing the Rework of Requirement Change. Canadian Conference on Electrical and Computer Engineering, Canada.
- Boehm, B. and R. Turner (2005). "Management Challenges to Implementing Agile Processes in Traditional Development Organizations." IEEE Software 22(5): 30-39.
- Bunse, C., R. L. Feldmann and J. Dörr (2004). Agile Methods in Software Engineering Education. Extreme Programming and Agile Processes in Software Engineering: 5th International Conference, XP 2004, Garmisch-Partenkirchen, Germany.
- Ceschi, M., A. Sillitti, G. Succi and S. De Panfilis (2005). "Project Management in Plan-Based and Agile Companies." IEEE Software 22(3): 21-27.
- Cohn, M. and D. Ford (2003). "Introducing an Agile Process to an Organization." IEEE Software 36(6): 74-78.
- Dagnino, A., K. Smiley, H. Srikanth, A. I. Antón and L. Williams (2004). Experiences in Applying Agile Software Development Practices in New Product Development. Proceedings of the 8th International Association of Science and Technology for Development, MIT, Cambridge, USA.
- Drobka, J., D. Noftz and R. Raghu (2004). "Piloting XP on four mission-critical projects." IEEE Software 21(6): 70-75.
- Dubinsky, Y. and O. Hazzan (2004). Roles in Agile Software Development. Extreme Programming and Agile Processes in Software Engineering: 5th International Conference, XP 2004, Garmisch-Partenkirchen.

REFERENCES - 2

- Dubinsky, Y., O. Hazzan and A. Keren (2005). Introducing Extreme Programming into a Software Project at the Israeli Air Force. 6th International Conference on Extreme Programming and Agile Processes in software Engineering, Sheffield, UK.
- George, B. and L. Williams (2002). An initial investigation of test driven development in industry. ACM Symposium on Applied Computing, Melbourne, Florida, USA.
- Gittins, R., J. Bass and S. Hope (2004). A Comparison of software Development Process Experiences. Extreme Programming and Agile Processes in Software Engineering: 5th International Conference, XP 2004, Garmisch-Partenkirchen, Germany.
- Grenning, J. (2001). "Launching extreme programming at a process-intensive company." IEEE Software 18(6): 27-33.
- Grosman, F., J. Bergin, D. Leip, S. Merritt and O. Gotel (2004). One XP Experience: Introducing Agile (XP) Software Development into a Culture that is Willing but not Ready. Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research, Markham, Ontario, Canada.
- Howard, D. (2003). Swimming Around the Waterfall: Introducing and Using Agile Development in a Data Centric, Traditional Software engineering Company. Extreme Programming and Agile Processes in Software Engineering: 4th International Conference, XP 2003, Genova, Italy.
- Karlström, D. (2002). Introducing Extreme Programming — An Experience Report. Proceedings of the 3rd International Conference on eXtreme Programming and Agile Processes in Software engineering, Alghero, Sardinia, Italy.
- Keefe, K. and M. Dick (2004). Using Extreme Programming in a capstone project. Proceedings of the sixth conference on Australian computing education, Dunedin, New Zealand.
- Kivi, J., D. Haydon, J. Hayes, R. Schneider and G. Succi (2000). Extreme programming: a university team design experience. Canadian Conference on Electrical and Computer Engineering, Halifax, Nova Scotia, Canada.
- Kähkönen, T. (2005). Life Cycle Model for Software Process Improvement Project Deploying an Agile Method. International Conference on Agility, Helsinki.
- Law, A. and R. Charron (2005). Effects of Agile Practices on Social Factors. Human and Social Factors of software Engineering, St. Louis, Missouri, USA.
- Lindvall, M., D. Muthig, A. Dagnino, C. Wallin, M. Stupperich, D. Kiefer, J. May and T. Kähkönen (2004). "Agile software development in large organizations." IEEE Software 37(12): 26-34.
- Lippert, M., P. Becker-Bechau, H. Breitling, J. Koch, A. Kornstädt, S. Roock, A. Schmolitzky, H. Wolf and H. Züllighoven (2003). "Developing Complex Projects Using XP with Extensions." IEEE Software 36(6): 67-73.
- Lytinen, K. and G. M. Rose (2005). "How Agile Is Enough? Towards A Theory of Agility in Software Development." Sprouts: Working Papers on Information Environments, Systems and Organizations 4(4): 169-192.

REFERENCES - 3

- Macias, F., M. Holcombe and M. Gheorghe (2003). Design-led & Design-less: One Experiment and Two Approaches. Extreme Programming and Agile Processes in Software Engineering: 4th International Conference, XP 2003, Genova, Italy.
- Mann, C. and F. Maurer (2005). A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction. Proceedings of XP/Agile Universe 2005, Denver Colorado, US.
- Maurer, F. and S. Martel (2002). On the Productivity of Agile Software Practices: An Industrial Case Study. 2002.
- McMahon, P. (2005). "Extending Agile Methods: A Distributed Project and Organizational Improvement Perspective." Crosstalk 18(5): 16-19.
- Morsicato, R. and M. Poppendieck (2002). "XP in Safety-Critical Environment." Cutter IT Journal 15.
- Mueller, G. and J. Borzuchowski (2002). Extreme Embedded: A Report from the Front Line. 17th Annual ACM Conference on Object Oriented Programming Systems Languages and Applications, Seattle, USA.
- Murru, O., R. Deias and G. Mugheddu (2003). "Assessing XP at a European Internet company." IEEE Software 20(3): 37-43.
- Müller, M. M. and O. Hagner (2002). Experiment about test-first programming. Conference on Empirical Assessment In Software Engineering (EASE), Keele, UK.
- Müller, M. M. and W. F. Tichy (2001). Case study: Extreme programming in university environment. International Conference on Software Engineering (ICSE23), Toronto, Canada.
- Nawrocki, J., B. Walter and A. Wojciechowski (2002). Comparison of CMM level 2 and eXtreme programming. 7th European Conference on Software Quality, Helsinki, Finland, Springer.
- Nawrocki, J. and A. Wojciechowski (2001). Experimental evaluation of pair programming. ESCOM 2001, London, UK, Shaker Publishing.
- Noble, J., S. Marshall, S. Marshall and R. Biddle (2004). Less Extreme Programming. Proceedings of the Sixth Conference on Australian Computing Education, Dunedin, New Zealand.
- Rasmusson, J. (2003). "Introducing XP into Greenfield Projects: Lessons Learned." IEEE Software 20(3): 21-28.
- Reichlmayr, T. (2003). An Agile Approach to an Undergraduate Software Engineering Course Project. 33rd ASEE/IEEE Frontiers in Education Conference, Boulder, CO.
- Reifer, D. (2002). "How Good Are Agile Methods?" IEEE Software 19(4): 16-18.
- Reifer, D. (2002). How to get the most out of Extreme Programming/Agile methods. XP/Agile Universe 2002, Chicago, Illinois, USA.
- Rising, L. and N. S. Janoff (2000). "The Scrum Software Process for Small Teams." IEEE Software 17(4): 26-32.

REFERENCES - 4

- Rostaher, M. and M. Hericko (2002). Tracking test first pair programming - an experiment. XP/Agile Universe 2002, Chicago, Illinois, USA, Springer-Verlag.
- Rumpe, B. and A. Schröder (2002). Quantitative Survey on Extreme Programming Projects. XP 2002, Alghero, Sardinia, Italy.
- Schuh, P. (2001). "Recovery, Redemption, and Extreme Programming." IEEE Software 18(6): 34-41.
- Schwaber, K. and M. Beedle (2002). Agile Software Development with Scrum, Prentice-Hall, Inc.
- Succi, G., W. Pedrycz, M. Marchesi and L. Williams (2002). Preliminary analysis of the effects of pair programming on job satisfaction. XP 2002, Alghero, Sardinia, Italy.
- Sutherland, J. (2001). "Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies." Cutter IT Journal 14(12): 5-11.
- Sutherland, J. (2004). AgileDevelopment: Lessons Learned from the First Scrum. Cutter Agile Project Management Advisory Service: Executive Update. 5: 1-4.
- Svensson, H. and M. Höst (2005). Introducing an Agile Process in a Software Maintenance and Evolution Organization. Proceedings of the 9th European Conference on software Maintenance and Reengineering, IEEE Computer Society.
- Turk, D., R. France and B. Rumpe (2005). "Assumptions Underlying Agile Software-Development Processes." Journal of Database Management 16(4): 62-87.
- Van Schoownderwoert, N. and R. Morsicato (2004). Taming the Embedded Tiger - Agile Test Tevhniques for Embedded Software. Agile Development Conference, Salt Lake City, USA.
- Williams, L. (2004). Extreme Programming Practices: What's on Top? Agile Software Development & Project Management Advisory Service. 5.
- Williams, L., R. R. Kessler, W. Cunningham and R. Jeffries (2000). "Strengthening the case for pair programming." IEEE Software 17(4): 19-25.
- Williams, L., W. Krebs, L. Layman, A. I. Antón and P. Abrahamsson (2004). Toward a Framework for Evaluating Extreme Programming. 8th Conference on Evaluation & Assessment in software Engineering, Edinburgh, UK.

THANK YOU!

Questions and comments?

Contact me at:

Pekka.Abrahamsson@vtt.fi

<http://agile.vtt.fi>

