Shared Experiences on Agile Methods
Results from eWorkshops

Mikael Lindvall
Fraunhofer Center – Maryland

and

Forrest Shull, Patricia Costa, Roseanne Tesoriero,
Vic Basili, Barry Boehm, Laurie Williams, Marvin Zelkowitz

plus 35+ participants and 50+ observers
Outline

- Motivation for the agile eWorkshop to collect empirical data
- What is an eWorkshop?
- Summary of expert “eWorkshop” to discuss agile topics:
  - Introducing agile methods
  - Documentation strategies in agile
  - Test-Driven Development
  - Feedback
  - Next Steps and other activities
Motivation for this Work

• Developers/managers need solid decision-support for adopting agile methods (from our survey among industry partners)
  – “Can I use the practice ‘out of the box’”? 
  – “What difficulties should I worry about?” 
  – “What kinds of projects fit best with agile?” 
  – “Does it scale up?”

• Agile proponents need help to overcome resistance
  – “Not Applicable Here?” Well, it was applicable somewhere like here…
  – …and how it happened.

• An empirical approach addresses these needs based on observation and evaluation, not theory.
Motivation for this Work

- However, empirical studies are just getting started
  - Not a lot of data collected yet
- At this stage, need to:
  - Understand which are the important tenets
  - Understand what empirical information already exists
  - Collect, abstract, and disseminate that information
  - Refine new, testable hypotheses of value to the community

- We executed three “eWorkshops” and discussed the results with industry partners
- We also applied agile methods to our own software development
The eWorkshop

• Meetings among experts is a classical way of creating and disseminating knowledge.
  – Understand where consensus exists, level of confidence in existing results

• But:
  – Experts are spread all over the world
  – Workshops are generally not captured for further analysis
  – Certain personalities often dominate a discussion

• To overcome these problems, we designed the concept of the eWorkshop
  – An on-line meeting,
  – Replaces some face-to-face workshop
  – Uses simple collaboration tools
Example of eWorkshop Control Room
1. Moderator wiped the screen
2. Moderator: Hello and welcome to the eWorkshop!
3. barry: I think the best we can do for this limited time period is to
define it as those which follow the 4 values and 12 principles in the Agile
Manifesto.
4. acockburn: 4 thanks barry, that'll save time, I concur
5. kea_Schwaber: 3+
6. BillKeh: 3+
7. Frank_Maurer: 3+
8. Hakan_and_Joel: 3+
9. Atif_Memon: 3-
10. acockburn: 3+
11. Don_Wells: 3+
12. Apurva: 3+
13. TimM: 3+
14. Scott_Ambler: 3+
15. Bill_Wood: 3+
16. Gary_Pollice: 3+
17. Moderator: Good, it looks like we agree on that definition.
18. Moderator: Let's add it to the whiteboard

8. What kinds of studies would you like to see?
   - Exploring the effectiveness of manual modeling techniques (CRC cards, white
     board sketches, low fidelity prototypes)
   - Experiments to measure how information about the system, the
     process, and technical skills are propagated through the team using
     agile practices
   - 499 acockburn: 474 Experiment design
      - I want to see quantitative studies comparing team morale / citizenship / 
        communication paths used, to project outcome. My prediction is that they link
        better than most others (other than the standard project killers). Without
        studying those factors, we only see the "Process", and the important factors 
        slip through the survey's net.

9. What are the success factors for agile development?
   - Negotiation: If people are willing to keep talking and learning, you are, 
     almost by definition, agile.
   - Ability to continuously learn
   - Responsibly responding to change.
   - Environment where rapid communication is enabled.
   - Team reasonably in charge of its own destiny.
   - People are valued over other factors
Expert participants from all over the world....
Expert participants

- Designers of agile methodologies
  - Scott Ambler (Agile Modeling)
  - Kent Beck (XP)
  - Ron Jeffries (XP)
  - Ken Schwaber (Scrum)

- Developers/team leads
  - Ken Auer (RoleModel SW)
  - Bil Kleb (NASA LaRC)
  - Tim Mackinnon (Connextra Ltd.)
  - Peter Hantos (Xerox)

- Developers/team leads
  - Aldo Dagnino, Karen Smiley, ABB Corp.
  - Peter Nelson, Steve Galea and Gregory Smith, Boeing
  - Jan-Peter von Hunnius, DaimlerChrysler
  - Kurt Schneider, DaimlerChrysler
  - Christian Wege, DaimlerChrysler
  - Tuomo Kahkonen, Nokia
  - Observers: Christina Wallin, Julien Le, ABB Corp.
Expert participants

• Consultants
  – Don Wells (at DaimlerChrysler)
  – James Grenning, Object Mentor Inc., signatory of the Agile Manifesto
  – Narti Kitiyakara, NOLA Computer Services, Inc.
  – Donald Reifer, RCI and University of Southern California
  – Walker Royce, Gary Pollice, Paer Jansson, Rational Software
  – William Krebs, IBM Software Group
  – Randy Miller, TogetherSoft, XP Agile Universe
  – Richard Turner, DoD
  – William A. Wood, NASA LaRC, XP Agile Universe
Expert participants

• Researchers
  – Hakan Erdogmus (NRC) &
  – Joel Martin (NRC)
  – Frank Maurer (U. Calgary)
  – Pekka Abrahamsson, VTT Electronics Finland
  – Vic Basili, UMD and FC-MD
  – Winsor Brown - University of Southern California
  – Patricia Costa, FC-MD
  – Barry Boehm, University of Southern California, XP Agile Universe
  – Atif Memon, UMD and FC-MD
  – David Stotts, University of North Carolina
  – Giancarlo Succi, Barbara Russo and Michela, Free University of Bozen, Italy
  – Laurie Williams, North Carolina State University
  – Marvin Zelkowitz, UMD and FC-MD
Expert participants

- In the control room at FC-MD
  - Barry Boehm (Virtually, Lead discussant)
  - Vic Basili (Lead discussant)
  - Laurie Williams (Lead discussant)

- Marvin Zelkowitz (Room captain)
  - Patricia Costa (Follow-up)
  - Forrest Shull (Scribe)
  - Mikael Lindvall (Moderator)
Introducing Agile: Where to start? All or nothing?

A range of possible paths...

- **Introducing all practices works best in projects that starts from scratch** (Grenning, Wood)
- **Teams with legacy issues may have to take a more incremental approach** (Grenning)
- **Select practices that**
  - **Provide clear benefits, enable agility, are natural** (Erdogmus, Krebs)
  - **Do not depend on other practices** (Hunnius)
  - **Team finds most convincing/interesting/suitable** (Abrahamsson)

Selecting practices is not a technical but a motivational issue (Basili, Boehm)
Which practices are easy…

- Coding standards and code ownership are natural and easy to get acceptance for
- It has been easy to see the benefits from Automated Testing

…hard to get people interested in?

- Pair programming is hard to get acceptance for, until developers try it
- Many participants reported that Test First is beneficial, but difficult to introduce
- Metaphor is not a simple concept
- A survey conducted by Don Reifer found that most do not follow the 40 Hour Work Week rule
What is the essence of agile?

The essence of agile is Short Cycles (2-4w.) (Vote 100% 14/14) ...with customer feedback (9 participants)

Customer feedback is one of the major reasons for Short Cycles (Krebs)

If it does not include the customer, it is not agile (Wege)

Customer involvement is definitely high-payoff (Boehm), but it can be hard to get the customer involved, if you can identify a customer at all (Hunnius)

In large organizations, seldom only one customer (Abrahamsson, Turner)

If the customer is not definable, agile is probably not appropriate (Turner)
What if you cannot identify a “real” customer?

Projects can experience a lot of frustration without a customer (Wood, Schneider)

...assign a customer proxy!

...someone that plays the customer role to focus the project

Planning Game can still be done with proxies, but less effectively (Turner)

The customer is a surrogate and role-playing is predominate

...getting the customer to play seems to be the hardest thing experienced (Reifer)
What are required characteristics of customers (proxies)?

- **Business competence in the development target domains** (Abrahamsson)
- **A vision of where the project/product needs to go** (Grenning)
- **CRACK capabilities** (Boehm)

CRACK =

- Collaborative,
- Representative,
- Authorized,
- Committed,
- Knowledgeable.
Many paths proposed...

- A small set that has been proved to be useful is: Story Cards and the Planning Game (Schneider)
- Start with Incremental and Iterative Development and add other practices later (Wege)
- Divide the practices into three sets: (Krebs)
  1. All (but Refactoring, Pair Programming, and Test First)
  2. Pair Programming and Test First
  3. Refactoring
Teams interested in adopting agile should…

Look for danger signs, which may mean a particular project is not suitable for agile methods:

E.g. If a “customer” is not definable, agile is probably not appropriate

If you cannot adopt all practices at once, begin somewhere:

E.g. begin with Customer Involvement, Automated Test, Pair Programming, or Iteration/Short Cycle times

Critical Success Factors:

Whichever path you pick, implement “Short Cycles” soon
Make sure you have and involve a customer or a proxy
Evaluate the customer or proxy for suitability, e.g. CRACK
Documentation: What do you document?

Depends on many factors:

- The stakeholders’ needs

- The goals of the documents (Grenning, Turner, Hunnius)

Agile relies on tacit knowledge, what gets documented is a major issue (Abrahamsson)

Agile views documentation as non-productive (Reifer)

Challenge if the document is really needed:

- Write it only if it is needed

- Do not write it if objectives can be met in any other way (Krebs)

Look at what you did last time:

- If the documents were not used, do not write them again (Grenning)
Documentation: What do you document?

Anything that uses programmer time should be accounted for in the Planning Game (Grenning)

So, document needs as user stories, assign costs and let the stakeholders decide (Costa)
Agile and Documentation

Agile people document what they did, not what they planned to do (Williams, Grenning)

It is better to use working code as the measure of progress, not documents (Grenning) (Yes (14/18) 78%)

Agile focuses on code and tests (Reifer)

Executable documents = self-documenting code & test (Williams)

Self-documenting = Self-describing names, comments only when needed (Hunnius)

Executable documentation -- good code and test cases -- is better than big documents (Boehm) (Yes (15/18) 83%)
What is the experience with executable documentation?

Has led to smooth transitions to different team:

One example is from an open source environment. After about 2 weeks the speed was "normal" again. After a major Refactoring, the speed was way up.

Size: 50,000 lines of C.
Refactoring: Transition to C++, one person-month (Hunnius)

Another good experience was when a new team found executable documentation and high-level conceptual documentation useful.
Size: 50+ use cases (Smith)
How useful are documents in project transitions?

Much money is spent on documentation, but most of the documentation is shelf ware and not used by anyone (Wege)

… the new team rarely refers to documents, as they are often out of date (Krebs)

Updating documents never get to the top of the list (Krebs)

The money runs out before documents get written (Wood)

Main problem with non-executable documentation: No assurance that is in synch with the code (Erdogmus)
Teams should not produce documentation without:

- Identifying stakeholders for each document

- Explicitly assigning costs for the document, so that everybody can understand the tradeoffs involved

- Making sure that it is really needed and the last time you developed this type of document, it was really used

Teams are recommended to use “executable documentation:” They are the only documentation that is guaranteed to be 100% accurate in reflecting what was done, not what was planned.
Test-Driven Development (TDD)

TDD is the most productive practice (Wood, Krebs)
It is more predictable than "debug later" and less costly (Grenning)

It has been fairly easy to get the team to try TDD (Williams)
No, it is not easy to get a team to do it (Erdogmus, Wood, Hunnius)

Why is it so hard:
- It is such a big mind shift (Krebs)
- It is not easy to master (Erdogmus)
- It is hard to explain the business value of test code (Wege)

Some people love it (Reifer, Hunnius):
- They do it easily once a framework has been developed (Reifer)
- It gives them the structure they need to work under (Reifer)
How important is it to refactor your test code?

It is very important (Krebs, Dagnino, Hunnius)

...because it is the programmer documentation (Grenning)

...and must be readable, so Refactor it when needed (Hunnius)

Refactoring has been neglected and later regretted (Grenning)

Test embodies requirements, must change accordingly (Wood)

Test code often has a simple design and is easier to refactor

Keeping test code consistent with production code is not simple (Basili)

It is not like keeping in synch with documentation. When all tests are running, at least you know that something is right. When they are not, either you have to fix tests or code (Erdogmus)
How much test code is typically produced?

For every line of production code there will be about 1-2 lines of test code (Hunnius, Williams, Erdogmus)

Debug code (asserts mainly) adds another line per line of production code (Hunnius)

When test code gets refactored, the ratio is expected to increase (Erdogmus)

Probably very little test code in non-agile projects (test never) (Williams)

For non-agile medium-assurance software, 1:1 is a good ratio (Boehm)

For non-agile high-assurance software, the ratio can be 5:1 (Boehm)
TDD is promising, but there are some inhibitors to TDD …

TDD can be hard to introduce because it is such a radical shift in thinking

How can we measure the effectiveness of TDD?
Aiming to achieve 100% coverage is NOT the answer.
There are some guidelines as to the size of test code to be expected for programs of different size

Organizations should remember that TDD imposes its own overhead: Test code can impose significant maintenance costs
Feedback

590 Grenning: thanks. no crashes this time. good work!
593 Reifer: It was fun - thanks
596 Krebs: Outstanding! Great tool, great minds
598 Erdogmus: Best one ever I think! The rhythm was much better.
602 Boehm: Thanks, everybody! Lots of valuable information and insights!
604 Hunnius: Less "religious" than the last one
605 Dagnino: Thank you and this was very good
Feedback

Was this eWorkshop of value to you?
Yes (16/17) 94%
No (0/17) 0%
Not Sure (1/17) 6%

Would you recommend a colleague to participate in a future eWorkshop?
Yes (14/15) 93%
No (0/15) 0%
Not Sure (1/15) 7%

Did the eWorkshop tool meet the needs of this meeting?
Yes (10/10) 100%
No (0/10) 0%
Not Sure (0/10) 0%
Next steps and Other activities

- Face-to-Face workshop “What We Have Learned about Agile Methods” at the Agile Development Conference 6/28 Salt Lake City, Utah, to follow up on the results
- An eWorkshop to wrap it up
- Other activities:
  - State-of-the-art/State-of-the-practice report on Agile
  - Test-First Experiment: Does TF result in higher quality?
    - Interest group (US, Canada, Italy, Germany)
  - Lessons Learned on XP