

Agile Software-Development & Tools

Softnet-Event, Innsbruck 08-May-2008

Werner Wild
EVOLUTION® Consulting
Innsbruck, San Francisco

Study - Chaos Report

- Average Cost Overrun: 214%
- Average Time Overrun: 239%
- Average Specified Features implemented: 74%

- Only 12% of all SW-Projects are on-time and on-budget !!!

Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

(Kent Beck, Alistair Cockburn, Martin Fowler et al., Feb 2001)

Agile Methods

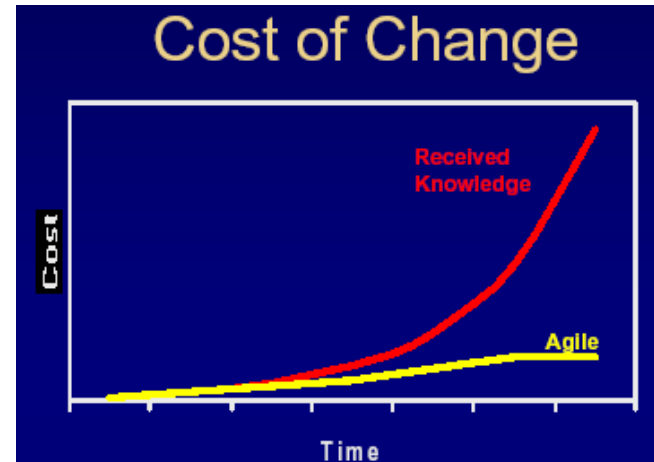
- eXtreme Programming (XP, Kent Beck)
- SCRUM (Ken Schwaber)
- Crystal (Alister Cockburn)
- ...

- Lean Development (Mary & Tom Poppendieck)

Changing the Mental Model



- Received Knowledge:
 - Die Change is Expensive
 - Don't Change Dies
- Taiichi Ohno
 - Economics Requires Many Dies Per Stamping Machine
 - One Minute Die Change*



- Received Knowledge:
 - Code Change is Expensive
 - Freeze Design Before Code
- The Agile Imperative
 - Economics Requires Frequent Change In Evolving Domains
 - Last Responsible Moment*

Principles of Lean Thinking

1. Eliminate Waste
2. Create Knowledge
3. Defer Commitment
4. Deliver Fast
5. Build Quality In
6. Respect People
7. Optimize the Whole

Principle 1: Eliminate Waste

- Taiichi Ohno on how the Toyota Production System works:

All we are doing is looking at the timeline from the moment a customer gives us an order to the point when we collect the cash. And we are reducing this timeline by removing the non value-adding wastes.

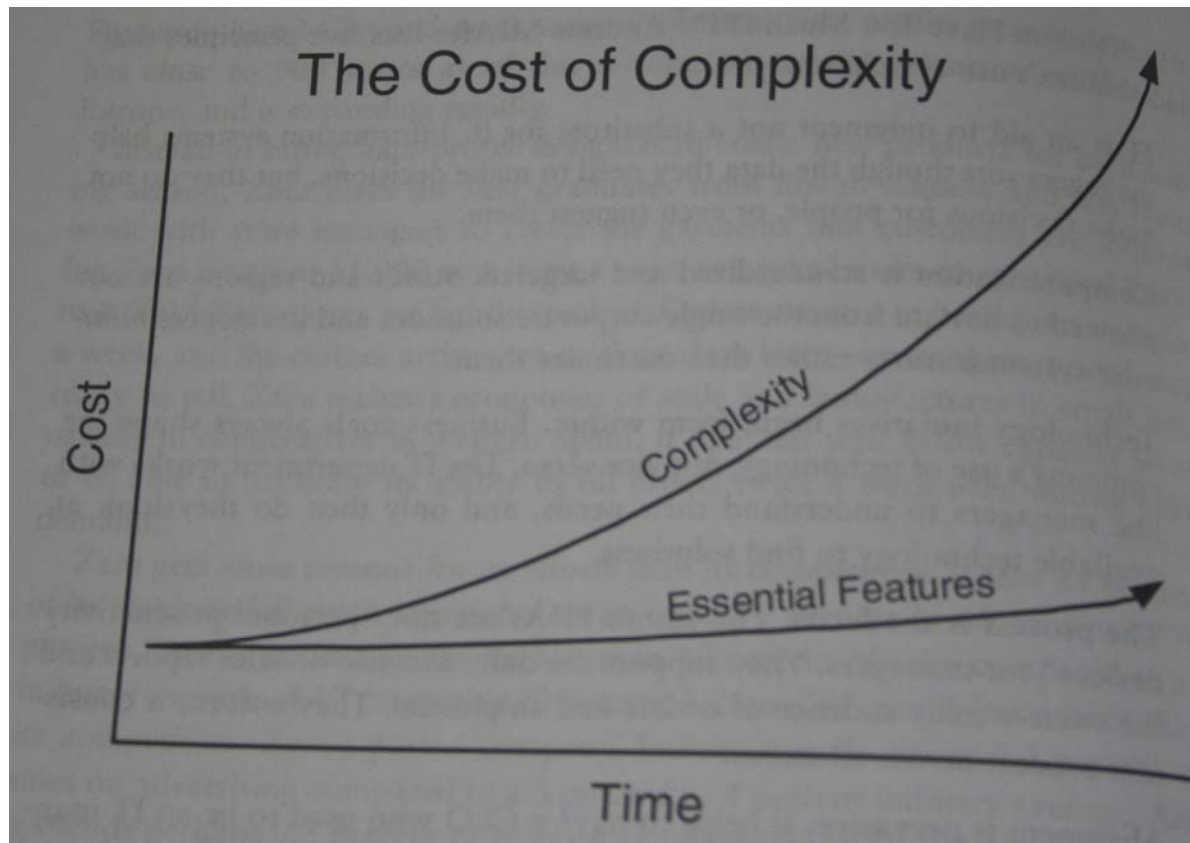
Eliminate Waste

- Waste
 - Anything that does not create value for the customer
 - The customer would be equally happy with the software without it

- Prime Directive of Lean Thinking
 - Create **Value** for the customer
 - Improve the **Value Stream** by removing non value-adding activities

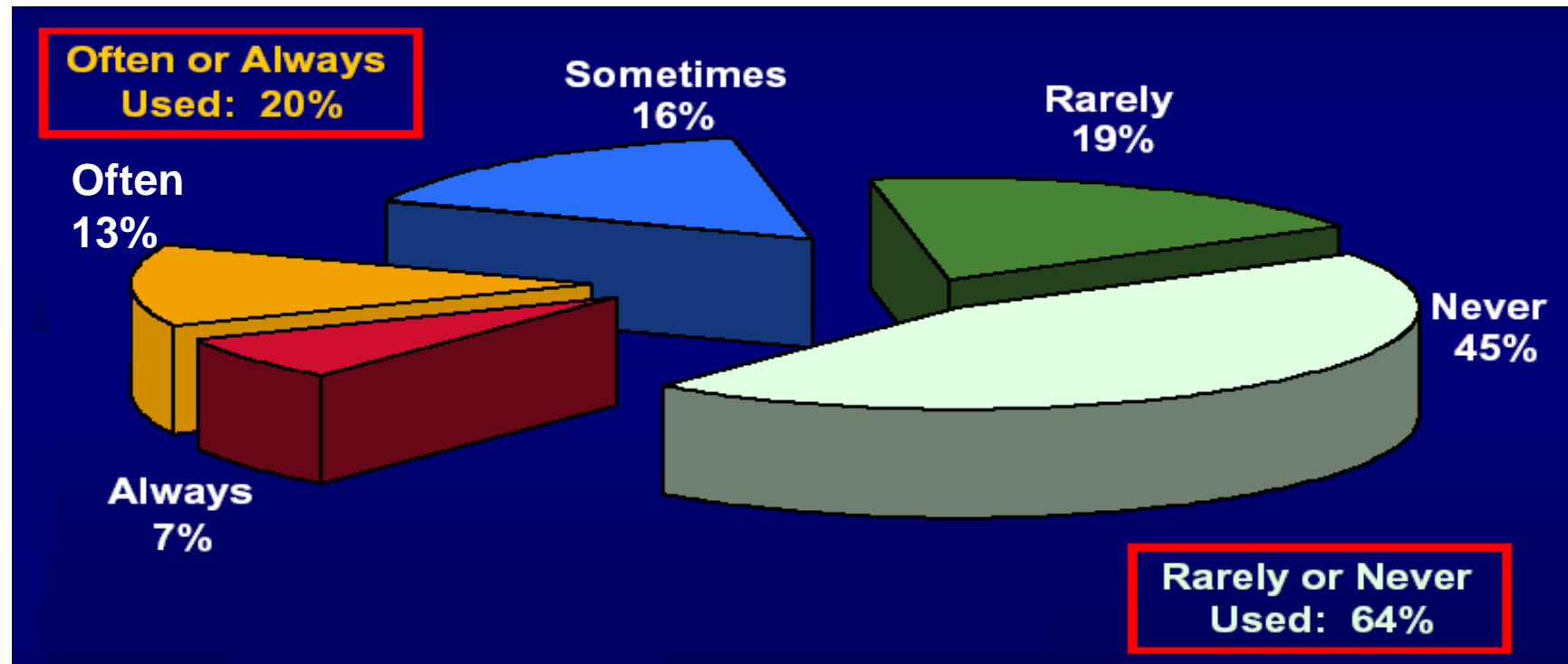
Complexity

- Complexity is the biggest source of waste



The biggest Source of Waste

Features and Functions Used in a Typical System

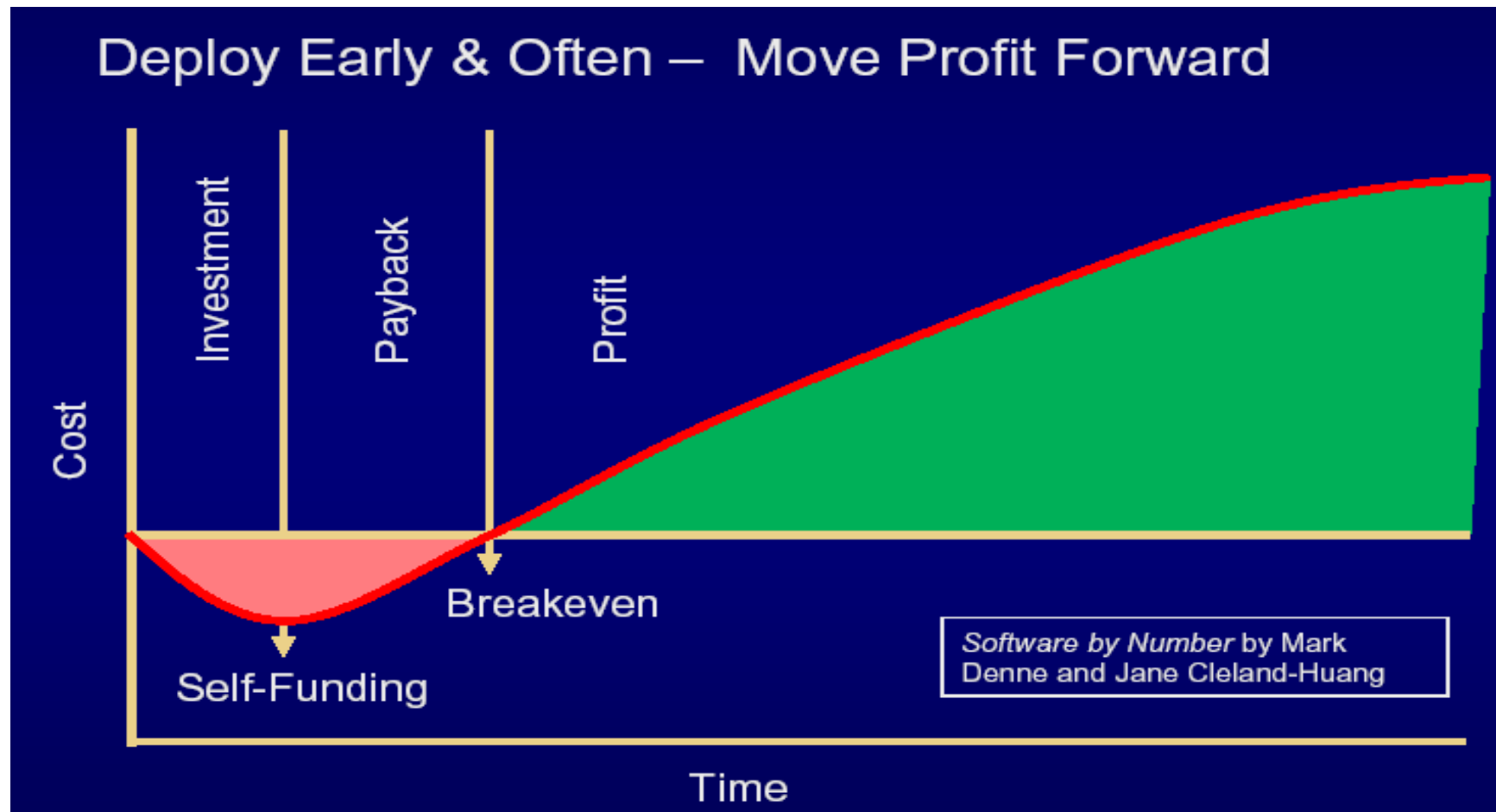


Standish Group Study Reported at XP2002 by Jim Johnson, Chairman

Write Less Code

- Justify every feature
 - Limit the features and the functions that make it into the code base
 - Every feature should prove that it will create more economic value than its lifecycle costs
- Minimum useful feature sets
 - Divide software into minimum useful features and deploy these one set at a time, highest priority first
 - Allows to use the software much faster

Minimum Useful Feature Sets



1st step is Seeing Waste

Manufacturing	Software Development
In-Process Inventory	Partially Done Work
Over-Production	Extra Features
Extra Processing	Relearning
Transportation	Handoffs
Motion	Task Switching
Waiting	Delays
Defects	Defects

eXtreme Programming

XP Values

- Communication
- Simplicity
- Feedback
- Courage
- Respect

XP – 12 Practices

- **The Planning Game**
Quickly determine the scope of the next release. Priorities
 - **Small releases**
Release new versions in very short cycles
 - **Metaphor**
Find a simple metaphor describing how the system works
 - **Simple design**
Make the design as simple as possible
 - **Testing**
Test the code **continuously**. Write the tests before the production code.
 - **Refactoring**
Mercilessly restructure the code to remove duplications, improve communication, simplify, or add flexibility
- **Pair programming**
Two programmers at one machine
 - **Collective ownership**
Everyone owns and can change any code anywhere in the system
 - **Continuous integration**
Integrate and build the system many times a day
 - **40-hour week**
Don't work more than 40 hours a week
 - **On-site customer**
Include real users in the team - they must speak with one voice though
 - **Coding standards**
Use a coding standard to improve communication

Tools

- Non-IT Tools
- IT based Tools

Non-IT Tools

- Index-Cards 😊
- Planning Poker Cards
- Kanban & Information Radiators
- Workspace
- Daily Retrospectives

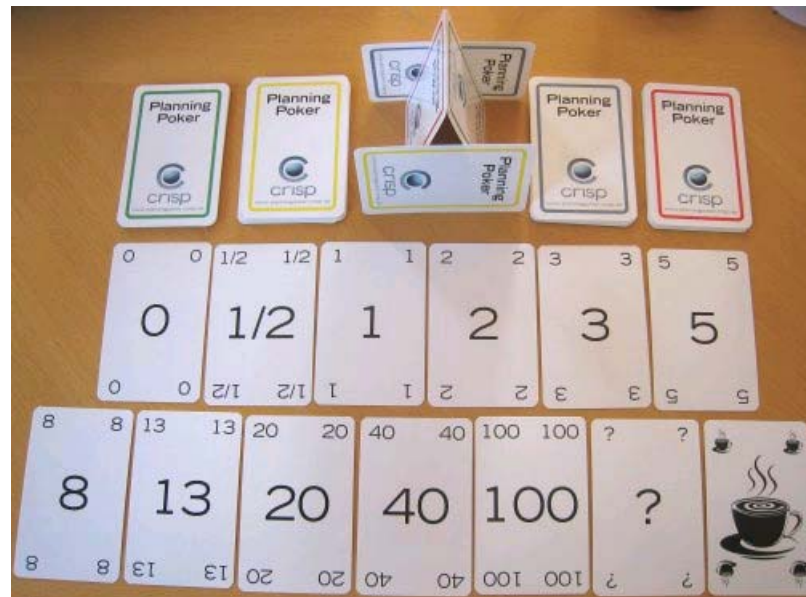
Index-Cards

- User Stories
- Acceptance Test (Stories, Scenarios)
- Planning & Scheduling tool
- Progress Indicator
- “Spike”
- Etc., ...

... just ... Don't leave home without them 😊 !!!

Planning Poker Cards

- Simple “Playing” Cards with Fibunacchi-like Numbers on them (by Mike Cohn)
- Used for effort estimation during the Planning Poker (weekly planning game)



Kanban & Information Radiators

- White Boards
- Flip Charts
- Cork Walls, Magnetic Walls
- Index Card Holders
- Large HDTV/LCD Screens

Goal:

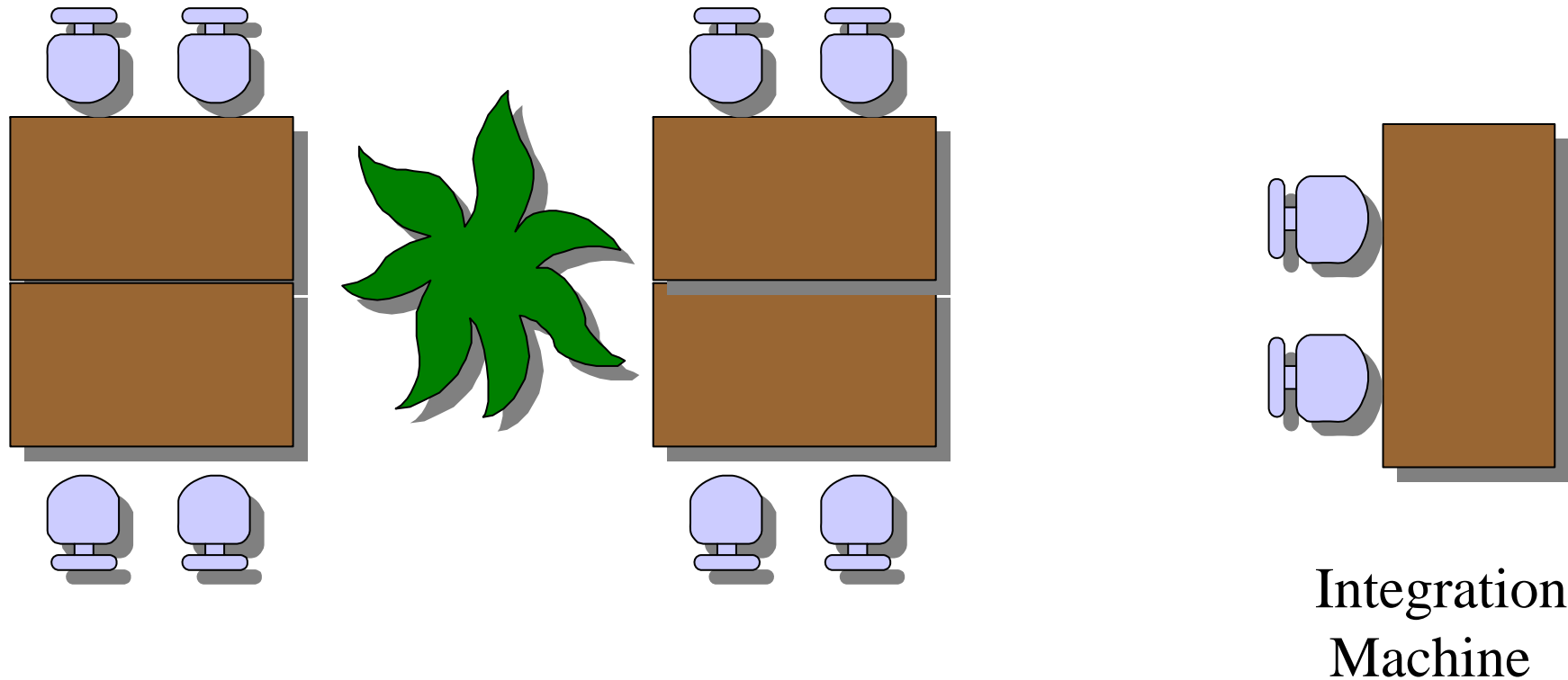
The overall status of the project and important, current details must be **immediately visible** when entering the development room !

There is nothing to hide on an Agile Project !!!

Office & Workspace

- Reduce Interruptions to emergencies only!
- Keep all developers in ONE room 😊
- Do Pair Programming (only) !
- Plaster walls with Information Radiators
- Have separate meeting rooms & private offices (can be shared)

Office Layout – Pair Programming



Office Layout - Machinery

- Pair Programming stations should be the fastest PCs you can get, with huge memory, and have at least two large (20"++), high dpi screens
- Integration machine for continuous integration (to do the actual integration work, plus for running end-user (acceptance) tests)
- “Stop the Line” device must be connected to the integration machine (e.g. lava-lamp)

IT-based Tools

- IDE
- Testing
- Integration & Deployment
- Information Documentation & Dissemination
- Communication
- Other Tools

IDE

- Fast & Responsive
- Incremental Compiling
- Automatic Build
- Must integrate support for:
 - Repository (CVS, Subversion, ...)
 - xUnit
 - Refactoring
 - Ant/Maven/ ...

Open Source Examples: Eclipse, Smalltalk

Testing

- Unit Testing: xUnit
- Acceptance Testing: FITnesse
- Performance Testing: TPTP in Eclipse
- Missing: great open source tool for GUI testing (!)

Integration & Deployment

- Continuous Integration: CruiseControl
- Deployment: Ant

Information Documentation & Dissemination

- Wiki
- Trac
- Bug tracking: bugzilla
- FITnesse

Communication

- VoIP: Skype, etc. (use headsets!)
- Video Conferencing/Webcam
- Chat: Skype, Windows Live, ICQ
- Screen-Sharing: VNC

Other Tools

Commercial:

- Mingle by Thoughtworks (Planning, etc.)
- TRICHORD by Change-Vision (Planning)

Conclusion

- Do not focus on IT based tools ! Use index cards, planning poker, information radiators
- Select an IDE which really “breaths” Agile (and hopefully is developed using Agile 😊)

Keep it simple !

Communicate, Communicate, Communicate!

Elicit & give ongoing, immediate Feedback !

Speak up!

Respect people !

... so “just” ...

Live the Agile Values !

Thanks for Your Attention !

Good Luck
for Your
Agile Endeavours !

Contact Information:

Werner Wild

EVOLUTION® Consulting

Jahnstr. 26

A-6020 Innsbruck

E-Mail: werner.wild@evolution.at

Literature

- Mary and Tom Poppendieck: *An Agile Toolkit for Software Development Managers*. Addison Wesley, 2003.
Mary and Tom Poppendieck: *Implementing Lean Software Development. From Concept to Cash*. Addison Wesley, 2006.
- Kent Beck: *Test Driven Development by Example* (Addison Wesley 2003)
Kent Beck: *eXtreme Programming Explained – Embrace Change* (Addison Wesley 2000)
Kent Beck: *Extreme Programming Explained: Embrace Change (2nd Edition)* (Addison Wesley 2004)
- Martin Fowler: *Refactoring – Improving the Design of Existing Code* (Addison Wesley 2000)
- Mark Denne, Jane Cleland-Huang: *Software by Numbers* (Prentice Hall 2004)
- Johannes Link: *Unit Testing in Java – How the Test Drives the Code* (Morgan Kaufmann 2003)
- Frank Westphal: *Testgetriebene Entwicklung mit Junit und FIT*(dpunkt Verlag, 2005)
- ... (Nearly) the complete Pragmatic Programmer's Series ...