Agenda

- Problem with traditional software engineering
- Why Agile is the solution?
- Roots of Agile
- Values of Agile
- Common implementations
  - Scrum
  - Kanban
- Common pitfalls
Me

- Born in Finland
- M.Sc. on Software Engineering from Tampere University of Technology
- ~10 years of experience
- Employed by Swisscom Strategy & Innovation
- Contact:
  - Twitter: @Pietrotull
  - kettunen@gmail.com
- Blog: www.pietrotull.com
Traditional S.E.

- Modelled after construction engineering
  - Architect
  - Software Architecture
  - Build tools
Nou Mestalla
The Problem

“Only thing that is constant is change”
- Heraclitus
Construction vs Software

low design cost & high build cost

vs

high design cost & minimal “build” cost
The Solution: Agile

- Adaptation
- Embrace change
"Agile methods are adaptive rather than predictive”
- Martin Fowler
Roots of Agile Movement

- MIT Hacker Culture
- Organizational Patterns by James O. Coplien
- Toyota Manufacturing System
- The New Product Development Game -1986
Agile Manifesto

- February 2001

- Kent Beck
- Mike Beedle
- Arie van Bennekum
- Alistair Cockburn
- Ward Cunningham
- Martin Fowler
- James Grenning
- Jim Highsmith

- Andrew Hunt
- Ron Jeffries
- Jon Kern
- Brian Marick
- Robert C. Martin
- Steve Mellor
- Ken Schwaber
- Jeff Sutherland
- Dave Thomas
Agile Manifesto

*Individuals and interactions over processes and tools*

*Working software over comprehensive documentation*

*Customer collaboration over contract negotiation*

*Responding to change over following a plan*
12 Principles of Agile

- customer satisfaction
- embrace change
- frequent delivery
- collaboration
- motivated individuals
- face to face
- working software
- sustainable development
- technical excellence
- simplicity
- self-organizing team
- retrospection
Holy Trinity of Software

- Lean Software Development
- Agile
- Software Craftsmanship
Three enemies of effectiveness

- **Muda**
  - waste
- **Mura**
  - inconsistencies
- **Muri**
  - disturbances in the flow
Lean applied to software

- Eliminate Waste
- Build Quality In
- Create Knowledge
- Defer Commitment
- Deliver Fast
- Respect People
- Optimize the Whole

by Mary Poppendieck
Software Craftsmanship

Not only working software,
   but also well-crafted software
Not only responding to change,
   but also steadily adding value
Not only individuals and interactions,
   but also a community of professionals
Not only customer collaboration,
   but also productive partnerships
Why Agile Works

- Cynefin framework
- Nature of exploration
- do - evaluate - adjust
Yin & Yang of Agile

- Technical Practises
- Process Management
Technical practices

- Pair programming
- Test driven development
- Continuous integration
- Continuous deployment
- Behaviour driven development / specification by example
Process Methodologies

- Scrum
- Kanban
- XP
Process Methods

- Just In Time decisions
- Pull -mechanism
- Visualization
- Transparency
- Splitting work into smaller pieces
- Limiting work in process
Scrum
Scrum
Scrum (early 90’s)
- Jeff Sutherland (Easel Corp)
- Ken Schwaber (Advanced Development Methods)

Jeff & Ken collaborated to present
- Scrum methodology at OOPSLA ’95
Scrum

- Agile process for producing business value
- Iterative
- Produce value
- does NOT prescribe technical practises
Scrum in action

Product Backlog → Sprint Backlog → Sprint → Working increment of the software

24 h
30 days
Scrum Roles

- Product owner
- Scrum master
- Team
Product Owner

- Decides priorities / order
- Vision of the product
ScrumMaster

- Takes care of the process
- Enables the team to do their work
- Owns the impediment list
Team

- Makes the magic happen
- Crossfunctional
  - includes all the skills to finish the product
- Self-organizing
Scrum Ceremonies

1. Sprint planning
2. Daily scrum meeting
3. Spring review
4. Spring retrospective
1. Sprint planning

Who

- Team, ScrumMaster & Product Owner

Agenda

- Discuss top priority backlog items
- Team selects which to do

Why

- Know what will be worked on
- Understand it enough to do it
2. The daily scrum

Parameters:
- Daily
- max 15 minutes
- standing

Not a problem solving meeting
- Whole world is invited
- only team, ScrumMaster, Product Owner can talk
2. The daily scrum

Questions

1. What did you do?
2. What will you do?
3. Is there anything stopping you? (impediment)
3. Sprint review

- Inspect and adapt the product
- Team presents what was accomplished
- Typically involves a demo
- Informal
4. Retrospective

- Inspect and adapt the process
- Everyone can participate
Scrum Artifacts

- Product backlog
- Sprint goal
- Sprint backlog
- Burndown chart
- Impediment List
Product backlog

- List of desired work
- Ordered / prioritized by the product owner
- Reorganized at the start of each sprint
Sprint goal

High level summary of where the focus is for given sprint

For the “high level” boss
Sprint backlog

- Evolves
- Team maintains
  - can add tasks
  - can remove tasks
  - re-estimate
- The team owns the sprint backlog
- “Best guess” what the team needs to do
- Progress visible in the task board
Burndown chart
Scaling Scrum

- Several Scrum teams
- Scrum of scrums
  - coordination over several Scrum teams
Kanban
Kanban

- Kan ban = “signal card”
- Originally by Taichi Ono (Toyota)
- Software Kanban by David Anderson
- Evolutionary approach
1. Start with what you have
2. Agree to pursue incremental, evolutionary change
3. Respect current process, roles & titles
4. Leadership at all levels
Kanban 6 practises

1. Visualize workflow
2. Limit work in progress
3. Manage flow
4. Explicit policies
5. Implement feedback loops
6. Improve collaboratively, evolve with experiments
1. Visualize workflow

- Analyze work states
- Define work item types
- Make problems visible
2. Limit work in progress

- Prevent multitasking / context switching
- Less work in progress = less waste
3. Manage flow

- Sustainable pace
4. Explicit policies

- No secrets
- Gew clear rules
5. Improve collaboratively

"To be termed scientific, a method of inquiry must be based on gathering empirical and measurable evidence subject to specific principles of reasoning”

- Isaac Newton

- Collaboration with all stakeholders
- avoid local optimizations
- Everything has it's own rhythm
Estimation

Lead time distribution (calendar days)

Occurrences vs. lead time in days
# Scrum vs Kanban

<table>
<thead>
<tr>
<th>Scrum</th>
<th>Kanban</th>
</tr>
</thead>
<tbody>
<tr>
<td>● WIP per sprint</td>
<td>● WIP per stage</td>
</tr>
<tr>
<td>● Sprint content set</td>
<td>● No untouchable tasklist</td>
</tr>
<tr>
<td>● Task size</td>
<td>● Task size unlimited</td>
</tr>
<tr>
<td>● Crossfunctional teams</td>
<td>● Allows specialist teams</td>
</tr>
<tr>
<td>● Timeboxed</td>
<td>● No time limits</td>
</tr>
<tr>
<td>● Velocity</td>
<td>● Lead time</td>
</tr>
<tr>
<td>● Fair amount of rules</td>
<td>● Very few rules</td>
</tr>
</tbody>
</table>
Tool for the job?
Agile Documentation

- Documentation is a poor substitute for conversation
- UI mockups
- Only code is up to date
public List<int[]> getThem() {
    List<int[]> list1 = new ArrayList<int[]>();
    for (int[] x : theList) {
        if (x[0] == 4) {
            list1.add(x);
        }
    }
    return list1;
}
public List<int[]> getFlaggedCells() {
    List<int[]> flaggedCells = new ArrayList<int[]>();
    for(int[] cell : gameBoard) {
        if(cell[STATUS_VALUE] == FLAGGED) {
            flaggedCells.add(cell)
        }
    }
    return flaggedCells;
}
Do the right thing
Do the right thing
User Stories

- Communication problem
- Way to describe functionality
- Collaborative effort
User Story example

As a <user> I want to <action>

E-Bank example

- As an account holder, I want to withdraw cash from ATM

Add context

- So that I can get money when the bank is closed
Given
The account balance is 100CHF and the card is valid

When
The account holder requests 20CHF

Then
The ATM should dispense 20CHF and the account balance should be 80CHF and the card should be returned
Minimum Viable Product

- pareto principle (80/20)
- Just In Time
<table>
<thead>
<tr>
<th>Feature</th>
<th>Nokia N95</th>
<th>Competitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>3G</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Camera</td>
<td>5mp</td>
<td>2mp</td>
</tr>
<tr>
<td>Memory card</td>
<td>microSD up to 32GB</td>
<td>no</td>
</tr>
<tr>
<td>MMS</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>3rd party apps</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Video out</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>VoIP</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Video calls</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Instant Messaging</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
N95 vs iPhone
Some other stuff
Holy Physical Task Board

- Promotes team interaction
- Visibility

ONLY ALAN WAS PREPARED TO ACKNOWLEDGE THE ELEPHANT IN THE ROOM.
Kaizen

- Continuous Improvement
Feedback

- feedback is information
- every mistake is a change to learn

“I didn’t fail, I found 10 000 ways that didn’t work”
- Thomas Edison
Testing

“Only fully tested software is the one not yet implemented.”

“Verify functionality”
Common Pitfalls

- Estimation
- Illusion of Importance
- Find and Replace
- Detailed plans
The Black Art of Estimation

“predicting is very difficult, especially if it involves the future”

- Niels Bohr