The Agile Mindset

Lean/Agile Workshop

~

Erik Philippus

© IMPROVEMENT BV
www.agile-architecting.com
erik.philippus@improvement-services.nl

Agenda

• The Origins of Lean Thinking
• Lean Software Development
• The Agile Approach

• Lunch

• Agile Scrum
• Cases
• Conclusions & Wrap Up

© IMPROVEMENT BV
www.agile-architecting.com
Agenda

- The Origins of Lean Thinking
- Lean Software Development
- The Agile Approach

- Lunch

- Agile Scrum
- Cases
- Conclusions & Wrap Up

The Arsenal of Venice

By 1400, the Arsenal of Venice was the world's most extensive industrial complex, with 3,000 employees and a production capacity of six galleys a month.

Venetian War Ship

The entrance of the Arsenal of Venice
The Arsenal of Venice

Over time the Venetians adopted a standardized design for the hundreds of galleys built each year.

The hull was completed first and then "flowed" past the assembly point for each item needed to complete the ship.

At the peak of its efficiency in the early 16th century, the Arsenal employed 16,000 people who could fit out, arm, and provision a galley in continuous flow, with standardized parts, going from start to finish in less than one day.

©Improvement BV www.agile-architecting.com

Hand-made Firearms

Until early 19th century, firearms were produced by enormous amount of handwork, making each firearm almost unique.

By 1765, the French general de Gribeauval had grasped the significance of standardized designs and interchangeable parts to facilitate battlefield repairs.

In 1799, Whitney introduced the concept of interchangeable parts for US firearms.

©Improvement BV www.agile-architecting.com
Perhaps the single most dramatic industry-changing moment in modern times was Henry Ford's revolutionary Model T and the moving assembly line.

Ford rearranged factory layouts to facilitate more efficient movement of people and materials.

Ford's Process Villages

- Fluctuations in end-customer demand buffered by vast bank of finished cars forced on the dealer network
- Congestions in upstream production buffered by large amounts of parts at every stage of production upstream from assembly

“Flow” production

“Mass” production
Synchronization & Rhythm

By the late 1930s, the German aircraft industry introduced **takt time** to synchronize aircraft assembly.

flexibility to make high-quality products in wide variety in small batches with short lead times.

Jidoka
imperfections drive positive evolution

自働化

* Automation

• Production is interrupted as soon as an imperfection is detected

• Any abnormality is directly solved and constitutes an opportunity for learning how to make things better

Jidoka is a pillar of the Toyota production system
• helps to detect problems earlier
• avoids spread of bad practices
• no production of defective products

Quality built-in to the process
Toyota Production System

Toyota was able to greatly reduce leadtime and cost, while improving quality at the same time.

Taiichi Onhno

• Build only what is needed
• Stop if something goes wrong
• Eliminate anything which does not add value
• Full utilization of worker’s capabilities

Taiichi Onhno

Toyota Production System

• Elimination of Waste (MUDA)

Muda has many forms, e.g. time, idle equipment, inventory. Most companies waste up to 70% of their available resources.

• Just in Time (JIT)

JIT is a ‘pull’ system of production: deliver on demand. Stock levels of raw materials, components, work in progress and finished goods can be kept to a minimum.
The Story of Lean Production

The concept of lean production is a process of value creation

- Lean focuses on eliminating waste in processes
- Lean is not about eliminating people
- Lean is about expanding capacity by reducing cost and shortening cycle times
- Lean is about understanding what is important for the customer

Lean Principles

Exercise:

- In small groups, discuss the following questions:

Which examples of lean production contain valuable lessons for system/software development?
Make a list of (potential) beneficial principles.

Which principles are (partially) being applied in your current project? What would be the benefits and pitfalls?

What do you think are the main blockades for adaptation of lean principles in your domain?
Agenda

- The Origins of Lean Thinking
- Lean Software Development
- The Agile Approach

- Lunch

- Agile Scrum
- Cases
- Conclusions & Wrap Up

Lean Software Development

principle 1: Eliminate Waste
  myth: Early specification reduces waste

principle 2: Build Quality In
  myth: The job of testing is to find defects

principle 3: Create Knowledge
  myth: Predictions create predictability

principle 4: Defer Commitment
  myth: Planning is Commitment

principle 5: Deliver Fast
  myth: Haste makes waste

principle 6: Respect People
  myth: There is one best way

principle 7: Optimize the Whole
  myth: Optimize by decomposition
Lean deals with value streams, not projects

A value stream goes from concept to cash

Typically, only a small part of lead-time is value-added activity

Value Stream

Exercise

- Make a rough outline of the value stream of your project. Address both value added activities and non-value added activities.

- What is approximately the ratio between these activities? Do you have suggestions for optimization of this ratio?
Elimination of Waste

Prime Directive of Lean Development:
- Create **Value** for the customer
- Improve the **Value Stream**

Elimination of Waste
- waiting
- defects
- unnecessary extras
- stock
- movement
- over production
- transport

Seeing Waste

**Exercise:**
Based on your own experience, give examples for potential sources of waste of system/software development
- waiting
- defects
- stock/inventory
- over-production
- movement/transport
- unnecessary extras
A Source of Waste

Features and Functions Used in a Typical System:

- Often: 13%
- Always: 7%
- Sometimes: 16%
- Rarely: 19%
- Never: 45%

Source: Standish Group Study Report

The Issue of Quality

Devil’s Triangle

Quality Costs More → Quality Saves

BUILD quality in, don’t try to TEST quality in

Software Quality is Free

It is not the quality that is expensive, but rather the lack of it
Principles of Effectivity

- **Use Small Batches**
  - Limit the amount of work in the pipeline
  - Minimize number of concurrent tasks

- **Minimize interrupts**
  - Stay focussed
  - No ad-hoc task-switching

- **Pull from customer demand**
  - Pull with an order
  - Don't push with a schedule

---

Defer Commitment

Make (irreversible) decisions at the last responsible moment in order to better accommodate change

change is inevitable - so deal with it!
Lean Principles
Resumé

Lean is about a culture of success and improvement

Agenda

• The Origins of Lean Thinking
• Lean Software Development
• The Agile Approach

• Lunch
• Agile Scrum
• Cases
• Conclusions & Wrap Up
The Era of Predictability

The 'McDonaldization' of software development

The Waterfall Approach

Most practitioner’s waterfall implementation lacks feedback loops

"We have decided to save the total screw-up part for the end".
Response to Unpredictability

<table>
<thead>
<tr>
<th>Predictive</th>
<th>Incremental</th>
<th>Adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waterfall</td>
<td>Spiral Model→RUP</td>
<td>Agile</td>
</tr>
<tr>
<td>Plan-driven</td>
<td>Minimizing risk</td>
<td>Change-driven</td>
</tr>
<tr>
<td>Decide early</td>
<td>response to unpredictability</td>
<td>Decide late</td>
</tr>
<tr>
<td>Deliver slow</td>
<td>Deliver fast</td>
<td></td>
</tr>
</tbody>
</table>

Response to Unpredictability

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

The Agile Alliance
Utah, February 2001
Agile Methods

- Evolutionary Development (EVO)
- Extreme Programming (XP)
- Crystal
- Dynamic Systems Development Method (DSDM)
- Test Driven Development (TDD)
- Feature Driven Development (FDD)
- Essential Unified Process
- Scrum

Why Agile?

<table>
<thead>
<tr>
<th>Reason</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerate Time-to-Market</td>
<td>22</td>
</tr>
<tr>
<td>Enhance Ability to Manage Changing Priorities</td>
<td>21</td>
</tr>
<tr>
<td>Increase Productivity</td>
<td>12</td>
</tr>
<tr>
<td>Enhance Software Quality</td>
<td>10</td>
</tr>
<tr>
<td>Improve Alignment Between IT and Business</td>
<td>9</td>
</tr>
<tr>
<td>Improve Project Visibility</td>
<td>6</td>
</tr>
<tr>
<td>Reduce Risk</td>
<td>6</td>
</tr>
<tr>
<td>Simplify Development Process</td>
<td>4</td>
</tr>
<tr>
<td>Improved/Increased Engineering Discipline</td>
<td>2</td>
</tr>
<tr>
<td>Reduce Cost</td>
<td>2</td>
</tr>
<tr>
<td>Enhance Software Maintainability/Extensibility</td>
<td>2</td>
</tr>
<tr>
<td>Improved Team Morale</td>
<td>1</td>
</tr>
</tbody>
</table>

source: 2008 survey under 3061 companies in 80 countries
Why Agile?

What are/were the organization's greatest concerns regarding the adoption of Agile development?

- Lack of upfront planning: 46%
- Loss of management control: 37%
- Lack of documentation: 36%
- Lack of predictability: 35%
- Lack of engineering discipline: 24%
- Inability to scale: 22%

(source: 2008 survey under 3061 companies in 80 countries)

Why Agile?

What are specific improvements you have actually realized by implementing Agile practices?

<table>
<thead>
<tr>
<th>% Improvement Realized</th>
<th>&gt;25%</th>
<th>25%</th>
<th>10%</th>
<th>NONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increased Productivity</td>
<td>25%</td>
<td>31%</td>
<td>33%</td>
<td>11%</td>
</tr>
<tr>
<td>Accelerated Time-to-Market</td>
<td>30%</td>
<td>26%</td>
<td>26%</td>
<td>17%</td>
</tr>
<tr>
<td>Reduced Software Defects</td>
<td>30%</td>
<td>25%</td>
<td>29%</td>
<td>16%</td>
</tr>
<tr>
<td>Reduced Cost</td>
<td>13%</td>
<td>17%</td>
<td>36%</td>
<td>35%</td>
</tr>
</tbody>
</table>

(source: 2008 survey under 3061 companies in 80 countries)
Agenda

• The Origins of Lean Thinking
• Lean Software Development
• The Agile Approach

• Lunch

• Agile Scrum
• Cases
• Conclusions & Wrap Up
What is Scrum?

Scrum is an Agile Project Management Technique

- iterative and incremental
- produces working software in short iterations
- adaptive rather than predictive
- people-oriented rather than process-oriented
- has a long history of success on a wide variety of projects, including game development
- is not a silver bullet

Scrum Basics

Scrum facilitates an incremental, feature-driven & time-boxed product realisation process
Scrum Basics

Team Commitment

What the team commits to, and what the product owner agrees to during sprint planning

should be delivered!
Reciprocal Commitment

The team commits to delivering the agreed-upon part of functionality

Management commits to leave priorities alone during the sprint

Scrum Roles

'Pig' roles:
- Product Owner
- Team
- Scrum Master

'Chicken' roles:
- Users
- Customers & Vendors
- Managers
Role of the Product Owner

The Product Owner:

- Understands the customer's needs and carries the product vision to the team
- Owner of the Product Backlog
- Defines the features of the product, decides on release date and content
- Priorizes features according to market value
- Can change features and priorities every sprint
- Accepts or rejects work results

Role of the Team

The Team:

- Understands the details of the Product Backlog
- Decides how much productive time it has available during the sprint
- Decides how many Product Backlog items it can commit to complete during the sprint
- Delivers potentially shippable product at the end of every sprint
- Organizes itself and its work
- Demos work results to the Product Owner
Role of the Scrum Master

The Scrum Master:

- Is the mentor and facilitator of the team and the moderator for Scrum meetings
- Takes care that the Agile/Scrum principles are understood and respected
- Shields the team from external interferences
- Enables close co-operation between all roles and functions
- Removes obstacles that impede team members
- Keeps track of the team progress

Sample Product Backlog

<table>
<thead>
<tr>
<th>Backlog item</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a player, I want the camera to stay oriented when it collides with bushes.</td>
<td>3</td>
</tr>
<tr>
<td>As a player, I want to perform a false compliance within the apprehend window to defeat the cop and avoid arrest</td>
<td>5</td>
</tr>
<tr>
<td>As a player, I want punches, reactions and blocks synchronized, so that fighting looks natural and realistic</td>
<td>3</td>
</tr>
<tr>
<td>As a player, I want to see a clear representative HUD that is art directed to match the game's universe.</td>
<td>8</td>
</tr>
<tr>
<td>As a CTO, I want an easy to use functional test framework so that programmers / testers can write tests.</td>
<td>5</td>
</tr>
<tr>
<td>As a player I want to see enemies get knocked down and get back up.</td>
<td>30</td>
</tr>
</tbody>
</table>
User Stories & Story Points

**User Stories:**

- Feature descriptions from anyone on the team or any customer
- Have "Conditions of Satisfaction" which can be tested in review
- Have no/minimal dependency on other stories
- Are estimated using 'Story Points'

**Story Points** are a relative measure of feature difficulty/complexity

Story Points are a more accurate measure of project velocity and release schedule than using hours and days.

---

Planning Poker

1. All the team members have a set of cards
2. Scrum Master reads description of the backlog Item
3. Everyone selects and simultaneously shows cards
4. If estimates vary significantly, high and low estimators briefly explain why they have estimated so
5. Repeat steps 3-4 until estimates stop converging
6. Decide estimate for backlog item
7. Move to next backlog item
As a player I want to see enemies have hit reactions when I attack them.

As a player I want enemies to have animated reactions when I hit them.

As a player I want enemies to have physical reactions when I hit them.

As a player I want there to be collision detection when I attack an enemy.

Detail is added as an item moves up the iceberg.

often by first splitting the backlog item.
Agile Architecting

The Product Backlog Iceberg

And then later, by adding 'Conditions of Satisfaction' to the backlog item

- An enemy twists left when hit on the left, and right when hit on the right
- The enemy staggers back when hit in the center
- The enemy tilts back when hit in the head

As a player I want enemies to have physical reactions when I hit them

Product Backlog

work items on a prioritized stack

- Features
- Defects
- Issues
- Customer Requests
- Market Demands
- Others

Each iteration implements the highest priority work items
New work items are prioritized and added to the stack
Work items may be reprioritized at any time
Work items may be removed at any time

high priority

low priority

high-priority items should be well-defined

low-priority items can be vague
Estimation of Productive Time

• task of 8 hours doesn't mean completion in 1 day
• incorporate time for: meetings, phone-calls, e-mails, bug-fixing, other responsibilities, etc.
• No contingency planning!

<table>
<thead>
<tr>
<th>Team Member</th>
<th>Available Days During Sprint</th>
<th>Available Hours Per Day</th>
<th>Total Available Hours During Sprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracy</td>
<td>8 days</td>
<td>4 hours</td>
<td>32 hours (=8 x 4)</td>
</tr>
<tr>
<td>Sanjay</td>
<td>7 days</td>
<td>5 hours</td>
<td>35 hours</td>
</tr>
<tr>
<td>Philip</td>
<td>8 days</td>
<td>4 hours</td>
<td>32 hours</td>
</tr>
<tr>
<td>Jing</td>
<td>6 days</td>
<td>5 hours</td>
<td>30 hours</td>
</tr>
</tbody>
</table>

Sample Sprint Backlog

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Mon</th>
<th>Tues</th>
<th>Wed</th>
<th>Thur</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create close punch animations for attackers</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create close punch animations for responders</td>
<td>16</td>
<td>12</td>
<td>16</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Adjust fight navigation</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>Fix camera bouncing off walls</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polish audio</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Tune attack percentage in AI</td>
<td>8</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Sprint Cycle

Product Backlog

Product Backlog Estimation
- Product Owner informs the team about the product vision
- Product Owner presents the items on the product backlog that needs to be estimated
- Team members estimate effort needed for realization of items

Sprint Preparation Meeting
- Product Owner & Team define the sprint goal
- Duration of sprint is decided
- Team member are allocated
- Scrum master makes inventory of total hours available for sprint
- Appointments & reservations for all meetings

Scrum Meetings (1)
Scrum Meetings (2)

**Sprint Planning Meeting**

- Product owner describes priorities
- Team breaks product backlog items (features) into tasks
- Team commits to agreed-upon amount of work

**Daily Scrum**

- Three questions:
  - What did you do yesterday?
  - What will you do today?
  - What's in your way?
- Commitment between peers, not a management status meeting!

Scrum Meetings (3)

**Sprint Review Meeting**

- Team demonstrates what is accomplished
- 2 hours preparation time
- No powerpoint
- Anyone can attend

**Sprint Retrospective**

- Focus on continuous improvement
- Look back at last sprint, and ask what we would like to:
  - Start Doing
  - Stop Doing
  - Continue Doing
Sprint Burn Down Chart

Optimistic estimation?

Pessimistic estimation?

Determination of Team Velocity

Project Burn Down Chart

Project X Burn Down Chart

Work Remaining

Time (days)

0 1 2 3 4 5 6 7 8 9 10 11 12

0 2000 4000

0 500 1000 1500 2000 2500 3000 3500 4000

0 1 2 3 4 5 6 7 8 9 10 11 12
The Human Factor

Increasing communication temperature is an important tenet of the Agile approach.
**Daily Scrum**

- Each team member answers the following 3 questions:
  - What did you do yesterday?
  - What will you do today?
  - What is in your way?

- Timeboxed: only 15 minutes!
- No more 'submarine behavior'

**Task Board**

<table>
<thead>
<tr>
<th>Story</th>
<th>To Do</th>
<th>In Process</th>
<th>To Verify</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Code the...</td>
<td>Code the...</td>
<td>Test the...</td>
<td>Code the...</td>
</tr>
<tr>
<td>As a user, I...</td>
<td>8 points</td>
<td>9</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Code the...</td>
<td>Code the...</td>
<td>Test the...</td>
<td>Test the...</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Test the...</td>
<td>Test the...</td>
<td>Test the...</td>
<td>Test the...</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

- As a user, I... | 5 points
- Code the... | 8
- Code the... | 8
- Test the... | 6
- Test the... | 6
For a multi-year product, break the project into a series of shorter interim (internal) releases.

3 months is a good horizon.
Scrum of Scrums
scaling Agile

Agenda

• The Origins of Lean Thinking
• Lean Software Development
• The Agile Approach

• Lunch

• Agile Scrum
• Cases
• Conclusions & Wrap Up
Exercise for cases

Split up in small groups, preferably with mixed levels of experience and expertise. With the described case as starting point, discuss the listed observations and consequences.

- Try to identify the underlying issue - what is the actual problem?
- What could be the (perceived) benefits of the current way of working?
- What forces (in the organization, in the team, etc.) could be responsible for continuation of the situation?
- How can an Agile way of working help to cure the described issue? What are potential benefits (and possible pitfalls) of the new approach?
- Give a short presentation of the highlights of the discussion in your group.

Case 1
Dealing with Change
Case 1
Dealing with Change

Case:
'Killer Features' - unplanned/last-minute changes

- Product Managers directly interacts with software developer(s) to propose ad-hoc modifications
- Software developer feels obliged to immediately implement the suggested modification

Symptoms:
- Frequent task-switching & disturbance
- Lack of focus in team
- Planning is in jeopardy, moving targets
- …

‘Walking on water and developing a product to specifications are both easy as long as both are frozen’

Case 2
Managing Expectations

MORE
FASTER
BETTER
SOONER
MORE
NOW!
Case 2
Managing Expectations

**Case:**
Expected outcome of assignments is not made explicit (enough)

- Co-existence of different perceptions/interpretations
- Inefficient feedback loops
- Late visibility of crucial product characteristics

**Symptoms:**
- No shared vision
- Wrong targets & wasted efforts
- Mind reading & submarine behavior
- Discontent & demotivation
- …

Case 3
Multidisciplinary Collaboration

San Gimignano
Case 3
Multidisciplinary Collaboration

Case:
Activities & disciplines are not well aligned

• Available expertise only partially utilized
• Team members unnecessarily go astray
• Late projects

Symptoms:
• Waiting
• Low morale
• …

Case 4
Managing Interrups
Case 4
Managing Interrupts

Case:
Interrupt-based arrangement of tasks

- Project leader puts pressure on developers to finish odd jobs
- No prioritization of tasks: everything is important
- Developer feels like ‘chief cook and bottle washer’

Symptoms:
- Fragmentation of work
- Little room for self-organization
- Long lists of impediments
- Delaying ‘regular’ work is common practice
- …

Agile Leadership

Delegates
Essentials

Interferes constantly

Delegates
Trivia

Allows subordinate initiatives
Agile is an adaptive approach to software development,
performed in a highly collaborative manner by self-organizing teams,
with just enough ceremony that produces high quality software in a cost effective and timely manner,
which meets the changing needs of its stakeholders.

Questions?