### Agile Thinking Petri Heiramo

Agile Coach, CST



#### What is Important in Agile?

# Values

- Principles
- Practices
- Rules

It is important to know Why things work so that we do not sabotage them (by accident).



#### **Agile Manifesto**

#### **Manifesto for Agile Software Development**

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.



Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently,

from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.



#### Agile Principles 2/2

## Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity – the art of maximizing the amount of work not done – is essential.

The best architectures, requirements, and designs emerge from selforganizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.



#### Iterative

- Plans based on previous iterations
  - What did we learn?
  - How have requirements changed since last iteration?
- Development process is adapted based on feedback
  - Practices, tools, communications, ...

A mini-project that includes work in most disciplines, ending in a stable executable.





#### **Playing Golf?**





www.collab.net





Copyright 2011 CollabNet Inc, and Petri Heiramo

www.collab.net

#### Incremental

- Each increment expands and extends functionality developed in the previous iteration
- Each increment delivers fully working functionality
  - Do not build castles on quicksand
- Functionality-driven requirements definition
  - Allows delivering complete user stories from each increment
  - Functionality must be split small enough to fit in one increment
- Incremental allows safe implementation as additional functionality is always built on top of working code



# The best bang-per-buck risk mitigation strategy we know is incremental delivery

- Tom DeMarco, 2003



#### **Risk and Value Driven**

- Focus on reducing risk in early iterations
  - Technical risk, e.g. performance, new technology...
  - Business risk, e.g. key business functionality, usability...
  - Architectural coverage, i.e. cover all key components with early features
- Prioritize features based on business value
  - Initially focus only on the most important ones
  - Focus on key functionality in each feature
  - Get early versions out to users to prove business value and gather feedback



#### **Time-Boxed**

- Never extend iteration length to fit scope during sprint
  - Adapt scope
- Try to keep a regular iteration length
  - Needed for velocity, estimation and planning
- During sprints, time, cost and quality are fixed
  - Features are the only flexible variable
- Also time-box releases
  - Communication with stakeholders and users
  - Prioritization of features



#### **People and Communication Driven**

#### Individuals and interactions over processes and tools

- Empower people and teams
  - Commitment and motivation
  - Decisions are made where the work is done
    - Best expertise to make them
- Facilitate communications-rich environment
  - Enable and encourage direct point-to-point and open many-to-many communication
  - Information radiators
  - Group collaboration techniques
  - Feedback and discussion





#### **Feature Teams**

- Traditional common approaches
  - Component teams, e.g. UI team, database team, engine team, ...
  - Functional teams, e.g. architecture team, development team, testing team, ...
- In this context,
  - Feature != subsystem, module, layer or component
  - Feature == *customer-centric* functionality
- Feature teams are customer needs driven
  - Multidiscipline, since building a full feature will require competences from many areas
  - Responsible for delivering all aspects of a feature
  - Minimizes the waste in hand-offs and context switches
  - The team sees the whole  $\rightarrow$  Quality



#### **Seeing Waste**

- Understanding and seeing waste is the first step to eliminating it
- Introduced by Lean in manufacturing environment
  - Translated to software development by Tom & Mary Poppendieck <sup>1</sup>
- "Maximizing work not done" & "Penny saved is a penny earned"
  - The most effective way to improve efficiency is to stop doing useless activities

#### Waste in Software Development

- Unnecessary Requirements & Gold-plating
- Work in Progress
- Extra Steps
- Finding Information
- Defects Not Caught by Tests
- Waiting, Including Customers
- Handoffs

<sup>1</sup> http://www.poppendieck.com/papers/LeanThinking.pdf

#### **Eliminating Unnecessary Features**





www.collab.net

#### **Optimized Value (ROI)**





www.collab.net

#### What If... Traditional Challenge





#### What If... Agile Opportunity #1





#### What If... Agile Opportunity #2





#### Obviously, ...

- Measuring and estimating value is difficult
  - Diagrams like I drew are very difficult in real life
  - Different business benefits behave differently
- Just remember:

# Success and value cannot be managed through costs



#### What Maximizes Value?

- User research and business analysis
- Competence and motivation of the people involved
- Amount of knowledge generated
  - Highly related with the amount of communication
- Ability to incorporate improvements
  - Highly related with technical quality of the system



#### **Key Challenges**

- Agility and Scrum is hard
  - Take it seriously, embrace its values, do also the hard stuff, play by the rules
- Knowing your customers' and users' real needs is hard
  - Understand what is valuable, then prioritize
- Getting the right people involved is hard
  - Get them to spend enough time with the project
  - Find the right suppliers



# Why Does It Work?



www.collab.net



www.collab.net

COLLABNE

#### Why Is Waterfall So Prevalent, Then?

#### A historical accident?

•There is no scientific evidence that waterfall works, and there never was

- CHAOS report 1995
  - 31% of projects cancelled before completion
  - 53% ran an average of +89% cost overrun
- •The original author was misunderstood
  - Winston Royce was, in fact, a proponent of iterative development
    - Waterfall works only in the simplest of cases, and even then, do it twice
- •The waterfall model was chosen for DOD-STD-2167
  - Even USA Department of Defence never succeeded with the model
    - 75% failed or were never used, only 2% were used without extensive modification
- •The DoD model was adopted in many other international standards
  - The DoD standard was later changed to support iterative, but the damage was done
  - The author of that standard would' ve chosen otherwise, given the chance to correct that mistake

•A lot of successful iterative development in 50' s - 70' s

•There is a culture of up-front, inflexible budgeting

#### Complexity



Source: Strategic Management and Organizational Dynamics by Ralph Stacey in Agile Software Development with Scrum by Ken Schwaber and Mike Beedle.



www.collab.net

Complex	Complicated
Retrospective coherence	Knowable causality
- software development	- traditional engineering
Chaotic	Simple
No causality	Known causality



#### **Managing Complexity**





Source: Strategic Management and Organizational Dynamics by Ralph Stacey

Copied from http://www.siliconyogi.com/andreas/

it\_professional/sol/complexsystems/StaceyMatrix.html



The question is not *"Are we Agile?"* 

## The question is "Are we delivering business value efficiently?"

#### Agility is a means to an end.

