Reflections about estimation, architecture, and agile issues

TDT4290 Customer Driven Project
October 16, 2017

frAGILE IS JUST A WORD
About me

2009 -

BEKK

2009 -

Lånekassen

2012 -

Safurudin Mahic

@ <3
The project work will be evaluated based on the quality of the project report and presentation delivered at the end of the course and the students' reflections on the project work.

How the group actually has worked, technical problems, customer behaviour and availability, etc. are a part of the reflections report.

The group is asked to deliver a 3-4 page report, reflecting upon their experiences during the project and reflecting upon what they had learned and how they could have done things differently. The focus of this part is on the process rather than the product.
Introduction

The following criteria are evaluated in an integrated way:

- Whether the group has solved the given assignment, according to the customer's objectives of the project.
- Team work efficiency and the team dynamics.
- Team work process improvement efforts.
- Reasonable grounds for decisions taken.
- Visibility of limitations done.
- The students' ability to reflect on the process during the project.
- Layout and structure readability.
- Logical flow in the report.

Making software is (unfortunately) not just about writing code.

It is equally much as to make informed decisions. This is why there are developed methods both in risk management, estimation, project management, architecture management.

Knowing about various principles, methodologies, patterns and good practices helps.

Think about how you can document and learn from your process.

Be aware of the choices you make – know why you make them.

Communication is the most important factor.
Project management
WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.

THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.

I'M GLAD IT HAS A NAME.

THAT WAS YOUR TRAINING.
Project refers to efforts made to achieve a defined goal, usually within a planned time and resource framework. The term is used for activities ranging from simple school projects to eg. engineering of complex oil platforms in the North Sea. As a tool for keeping projects within the framework given while achieving the goal, theories and methods have been developed for how to implement them in the best possible way.
Project management is about how to apply good practices in the fields of

- Process
- Estimation
- Excecution
- Architecture
- Many others

...to achieve a goal

“Project management is “the application of knowledge, skills, tools, and techniques to project activities in order to meet project requirements”

(PMI*, Project Management Body of Knowledge (PMBOK® Guide))

*The Project Management Institute (PMI) is an international professional society.
**BUSINESS / CUSTOMER & IT MUST COOPERATE**

<table>
<thead>
<tr>
<th>Business</th>
<th>IT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clarify expectations to each other</td>
<td>Visualise opportunities</td>
</tr>
<tr>
<td>Building competence and understanding of each other's areas</td>
<td>Turn business needs into good solutions</td>
</tr>
<tr>
<td>Establish funding that provides flexibility and ability to adapt</td>
<td></td>
</tr>
</tbody>
</table>

Provide direction and focus

Participate in discussions and make informed decisions - even when effects and execution requirements are uncertain

See opportunities and formulate needs
There are no silver bullets
Every single project is 

UNIQUE
**agile**

ˈadʒɪl/

*adjective*

able to move quickly and easily.
"Ruth was as agile as a monkey"

relating to or denoting a method of project management, used especially for software development, that is characterized by the division of tasks into short phases of work and frequent reassessment and adaptation of plans.
"agile methods replace high-level design with frequent redesign"
WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.

THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.

I'M GLAD IT HAS A NAME.

THAT WAS YOUR TRAINING.
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.
Starting next week, our meetings will be "stand-ups" with no chairs, so we'll be more focused.

So you examined all of the problems in the company and decided the root cause was chairs?

We're also loosening the dress code.

So our problems are chairs and pants?
Extreme Programming

Scrum

Lean

Crystal

DSDM

++
The Cost of Defects

Fix Earlier, reduce cost

Implementation
Unit Testing
Integration Testing
System Testing
In-service

http://www.embeddedinsights.com/channels/2012/03/19/unit-test-tools-and-automatic-test-generation/
Tip #1

Make sure you really understand the goal and the vision for the project.
Goal and vision for project

“The portal is supposed to be **self explanatory**, so that the users are **self-serviced** and **don’t need to contact Norwegian State Educational Fund**.”
“The best way to make sure your product meets the needs of your target audience is to expose your designs to the scrutiny of your users.”

Apple Human Interface Guidelines
How?
Customer/product owner

User

Why?
Not like this....

Like this!

Henrik Kniberg
Tip #2

Create an revise plan for the project
Plan Concert

- **Book Musicians**
  - Email Bands
  - Review Contract
  - Negotiation
  - Venue Contract Signed

- **Venue**
  - Find available venues
  - Visit each venue
  - Decide on Venue
  - Review Contract
  - Venue Contract Signed

- **Promotion**
  - Website
  - Email Blast
  - Radio Advertising
  - Facebook Ads

- **Tickets**
  - Setup Ticketmaster
  - Ticket Sales Start
  - Ticket Sales End

- **Concert**
  - Concert Night!
Generic Project Life Cycle

1. Define project goal
2. Plan project
3. Execute project plan
4. Close project
5. Evaluate project

Effort & Resources Required

Project Time Line

Start → Finish
Prosjektveiviseren

https://www.prosjektveiviseren.no/
Estimation
Two questions for you

• When do we estimate?  • Why do we estimate?
I need a budget estimate for my project, but I don't have a scope or a design for it yet.

Okay, my estimate is $3,583,729.

You don't know anything about my project. That makes two of us.
Project management triangle

- Cost
- Quality
- Scope
- Time

https://en.wikipedia.org/wiki/Project_management_triangle
The triangle refers to the “good – cheap – fast” dilemma. Too often, customers ask for a job of excellent quality, at an economical price, at a pace that requires warp speed. So the adage goes like this: “You can have two out of three - good, fast or cheap – but you can’t have all three.”
Project management triangle

35 user stories

Cost

Quality

Time

user stories
Project management triangle

35 user stories

3 million

Deadline – December 15
Project management triangle

35 user stories

3 million

Deadline – November 23
Waterfall
Plan driven
The plan creates cost / schedule estimates

Agile
Vision driven
The vision creates feature estimates

www.agile-scrum.be
I NEED A COST ESTIMATE ON YOUR PROJECT.
I HAVE NO IDEA. I HAVEN'T EVEN GATHERED THE USER REQUIREMENTS.
DON'T WORRY. I WON'T HOLD YOU TO THE ESTIMATE.
YES YOU WILL. YOU WILL PUT IT IN THE PLAN. FORGET WE HAD THIS CONVERSATION, AND FIRE ME WHEN I GO OVER BUDGET.

GIVE ME A NUMBER OR I'LL FIRE YOU RIGHT NOW.
OKAY, IT WILL COST TEN MILLION DOLLARS.
THAT'S TOO HIGH. IF YOU ALREADY KNOW THE COST, WHY ARE YOU ASKING ME?
SO YOU'LL FEEL LIKE YOU HAD INPUT. IS INPUT SUPPOSED TO FEEL THIS BAD?
“We know what we want. Can you estimate how long it will take to build?”

“We need to get these requirements nailed down before we can start development.”
Tip #3

Create an revise a scope and estimate for the project
I need a budget estimate for your project.

One billion dollars.

That doesn't sound reasonable.

I'll shout numbers and you can stop me when one sounds reasonable.

Please stop being you.

ELEVEN!
Why do businesses need cost estimates?

- To screen investment opportunities
- Plan expenditure cycles
- Obtain Financial sanction both internally and from any joint venture partners
- Budget, Control and Reporting on projects progress
  - Internally
  - Fiscal reporting

An estimate is a guess and you need to convince your audience that it is a good guess
Importance of Good Estimates
Correct Project Cost
Cost OverEstimate
• Project Cancelled- Lost economic opportunity
• Project proceeds – excess capital tied up

Underestimate
• Project proceeds
 o Capital overspend
 o Project economic criteria not achieved
 o Loss of confidence in company
 o Share/Stock price may be effected
 o Adversely effect financing of future investment opportunities
Estimate Key Components

An estimate should comprise of:

• A well worked base estimate
• A realistic contingency budget
• A credible accuracy range that reflects the project uncertainties and risks (threats and opportunities)

Definitions – Contingency / Management Reserve

Contingency is an integral part of the estimate: "An amount added to an estimate to allow for items, conditions, or events for which the state, occurrence, or effect is uncertain and that experience shows will likely result, in aggregate, in additional costs." (Association for the Advancement of Cost Engineering, AACE)

“The amount of funds, budget, or time needed above the estimate to reduce the risk of overruns of project objectives to a level acceptable to the organization.” (Project Management Body of Knowledge, PMBOK)

Establishing the Estimate

Technical Basis
• Process design
• Codes & standards
• Quantities

Execution Plan
• Contracting strategy
• Schedule
• Resources

Estimate Basis
• Unit rates
• Productivity assumption
• Owners management
• Exclusions
A typical classification of risks is based on the level of knowledge about a risk event's occurrence (either known or unknown) and the level of knowledge about its impact (either known or unknown). This leads to four possibilities (Cleden, 2009):

- **Known–knowns (knowledge)**
- **Unknown–knowns (impact is unknown but existence is known, i.e., untapped knowledge)**
- **Known–unknowns (risks)**, and
- **Unknown–unknowns (unfathomable uncertainty)**.

Normal Distribution Shape is determined by its mean ($\mu$) and standard deviation ($\sigma$). Probability is associated with area under the curve. Since the distribution is symmetrical, the following probability rules of thumb apply:

- About 68 percent of all the values will fall between $+1\sigma$ of the mean.
- About 95 percent of all the values will fall between $+2\sigma$ of the mean.
- About 99 percent of all the values will fall between $+3\sigma$ of the mean.
PERT distribution uses a three-point estimate where

- a denotes an optimistic estimate
- m denotes a most likely estimate
- b denotes a pessimistic estimate

**PERT Mean** = \( \frac{a + 4m + b}{6} \)

**PERT Standard Deviation** = \( \frac{b - a}{6} \)
Estimation techniques - PERT and Critical Path Method

**Optimistic**

**Most likely**

**Pessimistic**

**PERT weighed average**

\[
\frac{(a + 4m + b)}{6}
\]

**Contingency**

**Management reserve**

"A reserve fund for ‘unknown unknowns’ /risks. Handled by senior management – not included in the project baseline cost.” (PMBOK)

**Accuracy range**

\(+/- 2 \sigma \)

95% probability / confidence interval

http://www.pmknowledgecenter.com/node/59
<table>
<thead>
<tr>
<th>Task</th>
<th>Predecessors</th>
<th>a</th>
<th>m</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>none</td>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>1</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>0.25</td>
<td>0.5</td>
<td>3.75</td>
</tr>
<tr>
<td>D</td>
<td>B</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>E</td>
<td>B and C</td>
<td>1</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>F</td>
<td>D and E</td>
<td>1</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>6.25</td>
<td>13.5</td>
<td>47.75</td>
</tr>
</tbody>
</table>

\[ \sigma = 6.92 \]

**PERT weighted average** = 17.00

**95% confidence interval** = 30.83
Value and Waste

In assessing a process, it is important to understand what activities in the process actually add value to the end result. All other activities are wasteful.

**CVA** (Customer Value Added – or just VA for Value Added): adding form fit or function to a product or service, an activity that the customer would be willing to pay for in isolation if they knew it was being done – e.g. Creating code, implementing functionality.

**BVA** (Business Value Added – non-negotiable waste): an activity that is required to operate the business but the customer is unwilling to pay for – e.g. Budget tracking, code documentation.

**NVA** (Non-Value Added): an activity that is not required by the business nor is the customer willing to pay for – e.g. Waiting for resource allocation, requirements documents, estimation.
Principles of Agile Estimation

Agile estimation techniques are collaborative. All appropriate people are included in the process. For example the whole Scrum team participates in estimating effort of Product Backlog Items. Collaborative techniques are also designed so that it is impossible to blame someone for an incorrect estimate: there is no way to trace who estimated what.

Agile estimation techniques are designed to be fast ({-er than traditional techniques) and deliberately trade off accuracy. We are not trying to learn to predict the future or get better at estimation. Instead, we recognize that estimation is a non-value-added activity and minimize it as much as possible.

Most Agile estimation techniques use relative units. This means that we don’t try to estimate dollars or days directly. Instead, we use “points” or even qualitative labels and simply compare the items we are estimating to each other. This takes advantage of the human capacity to compare things to each other and avoids our difficulty in comparing something to an abstract concept (such as dollars or days).
<table>
<thead>
<tr>
<th>Discipline</th>
<th>Project</th>
<th>Development initiative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements analysis</td>
<td>27 %</td>
<td>15 %</td>
</tr>
<tr>
<td>Solution description</td>
<td>38 %</td>
<td>25 %</td>
</tr>
<tr>
<td>Construction</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>Acceptance</td>
<td>15 %</td>
<td>15 %</td>
</tr>
<tr>
<td>Admin</td>
<td>12 %</td>
<td>10 %</td>
</tr>
<tr>
<td>Adaptation</td>
<td>14 %</td>
<td>0 %</td>
</tr>
<tr>
<td>Technical support</td>
<td>9 %</td>
<td>9 %</td>
</tr>
<tr>
<td>Total</td>
<td>215 %</td>
<td>174 %</td>
</tr>
<tr>
<td>Discipline</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Requirements analysis</strong></td>
<td>Analysis and description of the functional scope of the project.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Detailing functionality</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product Owner time</td>
<td></td>
</tr>
<tr>
<td><strong>Solution description</strong></td>
<td>Technical description of the solution (2 steps)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Step 1 – detailed enough to give 3-point estimates</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Step 2 – detailed enough for development</td>
<td></td>
</tr>
<tr>
<td><strong>Construction</strong></td>
<td>System development + unit tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stabilized functionality by dedicated testers</td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance</strong></td>
<td>Test management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Plan system and acceptance tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Execute system and acceptance tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Document routines and procedures for operations</td>
<td></td>
</tr>
<tr>
<td><strong>Adaptation</strong></td>
<td>Internal training and communication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product verification</td>
<td></td>
</tr>
<tr>
<td><strong>Technical support</strong></td>
<td>Environment setup</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deploy setup</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test of deployment plan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Execution of deployment</td>
<td></td>
</tr>
<tr>
<td><strong>Admin</strong></td>
<td>Project management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Team management</td>
<td></td>
</tr>
</tbody>
</table>
Øker studiestøtten til elleve måneder

Regjeringen foreslår å utvide studiestøtten fra neste sommer. Det blir en ekstra måned med studielån, og totalt øker studiestøtten fra 94.000 kroner i år til 105.600 kroner. – Vi er veldig glade, sier NSO.
## Resource budget

<table>
<thead>
<tr>
<th>Person</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>37</th>
<th>38</th>
<th>39</th>
<th>40</th>
<th>41</th>
<th>42</th>
<th>43</th>
<th>44</th>
<th>45</th>
<th>46</th>
<th>47</th>
<th>#</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Budgeted</th>
<th>Actual</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>125</td>
<td>125</td>
<td>-15</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>125</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>125</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>125</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>125</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>125</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>125</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>125</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>100</td>
<td>-25</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>100</td>
<td>-25</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>100</td>
<td>-25</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>100</td>
<td>-25</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>100</td>
<td>-25</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>100</td>
<td>-25</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>125</td>
<td>-40</td>
</tr>
<tr>
<td></td>
<td>1600</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Rats, we aren't going as fast as we expected

Whoa! 5 new reports

Big innovation by Richard

Had to squash some bugs

Big push!
Tip #4

Track progress / speed / hit rate
Agile
AGILE

An iterative and incremental (evolutionary) approach to software development which is performed in a highly collaborative manner by self-organizing teams within an effective governance framework with “just enough” ceremony that produces high quality software in a cost effective and timely manner which meets the changing needs of its stakeholders.

Scott Ambler
Agile software development describes a set of values and principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams. It advocates adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change. These principles support the definition and continuing evolution of many software development methods.
Not like this....

Like this!
WHY AGILITY

• Need to more effectively respond to change – Organizational needs – Market demands – Threats and opportunities

• Manage evolutionary change to culture, products, and processes

• Frequently and incrementally deliver value to reach a desired goal or outcome
LEARN → BUILD → MEASURE → LEARN
Agility is the ability to both create and respond to change in order to profit in a turbulent business environment.

Agility is the ability to balance flexibility and stability.

BENEFITS

• Greater responsiveness
• Measured increase in productivity
• Lower costs
• Managed risk through greater visibility
• Increased customer satisfaction
• Better overall quality
• Improved team morale
Agile principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity - the art of maximizing the amount of work not done - is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
DO’S AND DON’TS

Agile methods do:
• Position us to be more responsive to changing priorities
• Allow better predictability of output for the foreseeable time horizon
• Make priorities and risk calculations more transparent

Agile methods don’t:
• Make engineers write more lines of code per day or designers create output faster
• Preclude the need to make good organizational decisions about product, priorities, and customer needs
• Allow the rest of the organization to assume they think they know what lies months ahead

In software development, sadly if you specify something, and everyone is doing their best, you’ll get what you want – at least what you specified. But, is it what you need?
AGILE MYTHS

• “Agile is the silver bullet for our organization!”
• “Agile doesn’t scale, so it can’t work for our organization!”
• “No more documentation or deadlines!”
• “Since we don’t do things upfront, we have no design and poor architecture!”
• “With so much rework going on, nothing new, and innovative gets worked on!”
• “Agile makes us faster!”

http://jpattonassociates.com/dont_know_what_i_want/
Perhaps you’ve been on this Agile project:
Customers meet with the team and successfully write a number of user stories. After a lot of conversation between developers and customers, developers estimate the stories. Customers prioritize them, highest value first, and choose the most important stories for the first release scheduled after six iterations.
Development starts, and things seem to go very well. In the fantasy world this story occurs in, all the development estimates were accurate. In the first couple iterations all scheduled stories are finished. But, that’s where things go wrong. After looking at the resulting software the customer says “Now that I see this, we’re missing a few things. And, although the things you’ve built meet the acceptance criteria, we, well.. uh… weren’t really sure about that acceptance criteria and now that we see it, it needs to change.”
“No problem” says the team. “Just write more stories. But, you’ll have to remove some of the others from this release in order to get them done on time.”
The customer’s shocked and angry. “What you’re saying is that I needed to get the requirements right up front! This smells just like waterfall – except without the up front time I’d need to even try to get the requirements right in the first place.”

http://alistair.cockburn.us/Three+cards+for+user+rights
http://jpattonassociates.com/dont_know_what_i_want/
Agile issues
I was visiting a relative a couple years ago. My poor cousin (the CEO of an insurance company) had been sold the Agile Silver Bullet™ and was pissed. He said something like:

It’s a sham! We changed the way we do everything. We brought in consultants. We hired these master project managers. And nothing worked! It made no difference. There’s no accountability. All I get is excuses.
Walls
Windows
Doors
Roof
Chimney
Lawn
Swimming pool
<table>
<thead>
<tr>
<th>Build</th>
<th>Roof</th>
<th>Wiring</th>
<th>Build Walls</th>
<th>Heating</th>
<th>Plumbing</th>
<th>Carpets</th>
<th>Decorate</th>
<th>Swim Pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>144</td>
<td>21</td>
<td>55</td>
<td>89</td>
<td>34</td>
<td>55</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

This story needs to go first.
Not like this....

1 2 3 4

Like this!

1 2 3 4 5
Would you drive a car on the road

... without proper training?

... without tyres?
Is not iterating
Flow efficiency

Work / Lead Time = Flow Efficiency (15% is "normal")
Unplanned work

Task switching

Value-Add

Initiative 2

Unplanned

(Production issues, maintenance, helping other teams, etc)

Not abnormal!
Benefits realization

"On time" is only part of risk
No learning / improvement

continuous improvement
Value and Waste

In assessing a process, it is important to understand what activities in the process actually add value to the end result. All other activities are wasteful.

**CVA** (Customer Value Added – or just VA for Value Added): adding form fit or function to a product or service, an activity that the customer would be willing to pay for in isolation if they knew it was being done – e.g. Creating code, implementing functionality.

**BVA** (Business Value Added – non-negotiable waste): an activity that is required to operate the business but the customer is unwilling to pay for – e.g. Budget tracking, code documentation.

**NVA** (Non-Value Added): an activity that is not required by the business nor is the customer willing to pay for – e.g. Waiting for resource allocation, requirements documents, estimation.
What’s the First Thing the Team Could Do?

I like to ask, “How little can we do?” Too often, the team has been asking “How much can we do?”

Sometimes, when we have stories, and we don’t know how to break them apart, we ask, “What’s the first thing that could add value?” Maybe that’s a good question here. You could ask, “What’s the first thing the team can do?” Or, “What’s the first thing we can do that would help us get to done in a one- or two-week iteration?” Or, “How do we finish a feature
The point of using agile is to get finish something valuable-to-the-business quickly, to get feedback. Why? For several reasons, but the first one is so you can change the project’s priorities. The second is so you can change the project portfolio. The third is to get feedback on what you’ve done. This is why every project needs to design its own way to get to done.

What happens when you have more unknowns? What if your organization is “addicted” to waterfall and long projects? When you can’t see the people on your project, because everyone isn’t in the same place? When you have a lot of technical risk, such as technical debt? How about when you’re starting a program and everyone is transitioning to agile (oh please, don’t do this). Or, you’re new to agile, and you don’t have training. I have a question: would you drive a car on the road with no training, either?
Definition of Done

When are you done with a feature?

*Getting the users’ feedback incorporated into the feature is part of the basic cost of developing the feature.*

<table>
<thead>
<tr>
<th>Started</th>
<th>In progress</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>Started</td>
<td>Ready for user review 1</td>
<td>Ready for user review 2</td>
</tr>
</tbody>
</table>

Selecting development strategies

- Some people like to get each story **as correct as possible** on the first round, and try to **minimize the changes** in rounds 2 and 3.
- Some people like to do the **minimum possible** on the first round, so they have the **most time to steer** according to the review feedback.

Acceptance criteria
<table>
<thead>
<tr>
<th>STORIES</th>
<th>URGENT! (High)</th>
<th>TO DO</th>
<th>NEXT</th>
<th>IN PROGRESS</th>
<th>DONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOGS / SUPPORT</td>
<td>~20%</td>
<td>16</td>
<td>6</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>ASYNC</td>
<td>10</td>
<td>11</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>STANDARD</td>
<td>10</td>
<td>16</td>
<td>20</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>OTHERS &amp; 50 (technical)</td>
<td>5</td>
<td>20</td>
<td>66</td>
<td>4</td>
<td>12</td>
</tr>
</tbody>
</table>

**Sprint 7**
09/01/13 → 30/01/13
3 Weeks to Release
(Total 1,814 Stories)
Factors For Successful Projects
User involvement
Executive management support
Clear statement of requirements
Proper planning
Realistic expectations
Smaller project milestones
Competent staff ownership
Clear vision & objectives
Hard-working, focused team

Engineering
Needs to have constant learning, refinement and adaptation to meet the environmental requirements
Architecture
What is software architecture?

Bass, Clements, and Kazman:

“

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.
Software architecture is a result of *business, technical & social* influences.
Questions

1. How do requirements lead to an architecture?
2. How do you analyze architectures?
3. How does architectures enable new organizational requirements and needs?
4. Is the architecture slowing you down?
Why software architecture?

1. Enable others to understand the system
2. Allow others to effectively work on parts of the system in isolation
3. Allows for extensions of the system
4. Facilitate reuse and reusability
5. Based on good principles and practices
High cohesion

Loose coupling
the Model-View-Controller architecture pattern

Model
contains business logic

Controller
interacts with Model to create data for the View

View
renders content to the user and relays user commands to the Controller
Typical EDA architecture

Event Producers
system  system  system

Event Transport

Event Bus

Event Consumers
system  system  system
What do we need to ask ourselves

1. Does the architecture fulfill the business goal?
2. What is the purpose of a diagram and who is the receiving end?
3. Does the architecture exhibit high cohesion and low coupling?
4. What known design patterns fit the problem at hand?
5. Does the architecture permit multiple teams to work on independent parts efficiently?
6. What positive and negative effects does the created architecture exhibit?
The system
Øker studie/Støtten til elleve måneder

Regjeringen foreslår å utvide studie/Støtten fra neste sommer. Det blir en ekstra måned med studielån, og totalt øker studie/Støtten fra 94.000 kroner i år til 105.600 kroner. – Vi er veldig glade, sier NSO.

VILDE FORNØYD – Detta er et historisk glødsmotiv for studentstøtten, sier leder i Norsk Studentorganisasjon Øla Magnusson Ryde.

Anne Lise Stranden

En ekstra måned: Det legges opp til elleve måneder utsnævningstøtte per undervisningsår til studenter og andre som får støtte etter samme regelverk. Første ekstra måned med støtte vil bli jørt 2015.
The Malleable Nature of Software

Evolution is more important in software than in other engineering disciplines.

Software engineering rarely involves “green field” development. Software needs to be constantly maintained and evolved to meet new business requirements. The cost incurred in evolution usually exceed the development cost by a factor of 3 or 4.
Conclusions?

The following criteria are evaluated in an integrated way:

- Whether the group has solved the given assignment, according to the customer's objectives of the project.
- Team work efficiency and the team dynamics
- Team work process improvement efforts
- Reasonable grounds for decisions taken.
- Visibility of limitations done.
- The students' ability to reflect on the process during the project.
- Layout and structure readability.
- Logical flow in the report.

Making software is not just about writing code.

It is equally much as to make informed decisions. This is why there are developed methods both in risk management, estimation, project management, architecture management.

Knowing about various principles, methodologies, patterns and good practices helps.

Think about how you can document and learn from your process.

Be aware of the choices you make – know why you make them.

It takes a great deal of effort to make agile work.

Agile cannot function in all circumstances – it is your job to identify whether the organization is mature enough to

Communication is the most important factor
Project retrospective

Also known as post mortem reviews

A common method to share knowledge within a development team.
Safurudin Mahic
safurudin.mahic@bekk.no