Agile Software Construction

Beatrice Åkerblom
beatrice@dsv.su.se

This Course

Course Contents

Software Engineering

Agile Methods

Course information

Daisy:

- Schedule
- Goals
- Requirements
- Important dates
- Grading criteria

http://dsv.su.se/~beatrice/AGILE

Lectures

Lectures can contain many other things than traditional teacher lecturing, e.g.

- Group assignments
- Paper reading
- Group discussions

Literature

1. “Software Engineering” Ian Sommerville
2. “Agile Software Development” - Alistair Cockburn
3. “The Pragmatic Programmer: From Journeyman to Master” - Hunt & Thomas

The books should preferably be read in this order and in different ways

Exam

Take-home exam

- Take-home exam distributed on Monday 2/3 8.00
- Hand-in of the take-home-exam Thursday 5/3 24.00

Project

The project will be a Scrum project

The goals are twofold

- Design a project that will make you learn the contents of the course
- Run the project as a Scrum project to get practical experience

Groups will be created using Daisy

- Subscribe to a group (avaliable from 19/1)
- Those who haven!t subscribed to a group on the 23/1 will be randomly assigned to a group

Requirements

Having successfully completed the course, the student should be able to:

describe what is the difference between plan-driven and agile processes and motivate the description

account for what is significant for a software engineering approach to software development and the principles behind it

account for what is significant for a plan-driven approach to software development and the principles behind it

account for what is significant for an agile approach to software development and the principles behind it

Requirements, cont!d

describe several techniques (both classical software engineering ones and agile ones) for:

- requirements elicitation and requirement management
- resource planning
- risk analysis and risk management
- test planning
- version management
- time and cost estimations
- project automation

make judgements about applicability of a these techniques in some known project, possibly combining plan-driven techniques with agile ones

Requirements, cont!d

decide and motivate why some technique is suitable for use in a plan driven software project

decide and motivate why some technique is suitable for use in an agile software project

find suitable software engineering methods for his or her work in future projects

easily adapt to work processes using software engineering techniques

“Course board”

The course board will meet with the course co-ordinator regularly during the course.

The role of the course board is to bring forward to the course co-ordinator any opinions about the course, contents or how things are done, from their fellow students.

If you as a student want to let the course co-ordinator know his or her opinion about anything on the course and don't want to contact her on your own, you can contact one of the members of the course board.

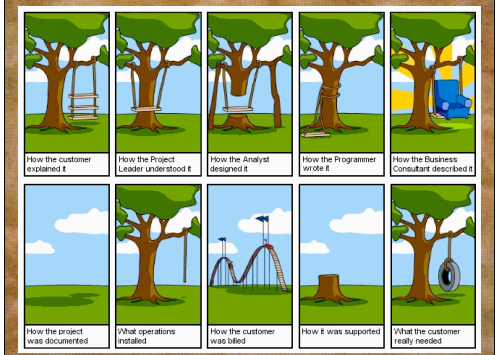
Introduction to Software Engineering and Agile Methods

The story of the $0.00 bill

A well-known story tells of a man who once received a $0.00 bill. He laughed at it and threw it away. When he received the reminder bill, he reacted the same way.

When he received the next bill claiming that legal actions would be taken if he did not immediately pay $0.00 he got worried and decided to mail a check for $0.00. This had the desired effect and a few days later he received the receipt for $0.00.

A few days later the man got a phone-call from his bank asking if he had paid someone with a $0.00 check. The man agreed and told the whole story. When he had finished, the bankwoman didn't laugh, but quietly asked “Have you any idea what your check for $0.00 did to our computer system?”.



60 years of Software Engineering

1940s: First computer users wrote machine code by hand.

1950s: Early tools; macro assemblers and interpreters First generation optimising compilers.

60 years of Software Engineering (cont!d)

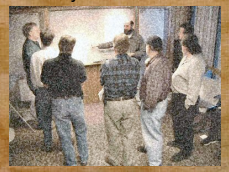
1960s: Second generation tools -- optimising compilers and inspections. First really big projects. Commercial mainframes and software for big business.

1970s: Collaborative software tools; Unix, code repositories, make, etc. Minicomputers and small business software.

60 years of Software Engineering (cont!d)

1980s: Personal computers and workstations. Emphasis on processes. The rise of consumer software.

1990s: Object-oriented programming and agile processes. WWW and software everywhere.



Scope of Software Engineering

Historical Aspects

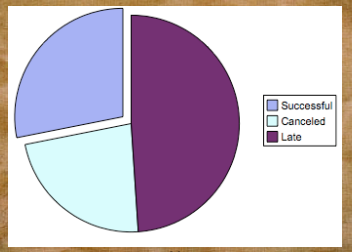
- 1968 NATO Conference, Garmisch
- Aim: to solve the “Software Crisis”
- Software is delivered
  - Late
  - Over budget
  - With residual faults

Background

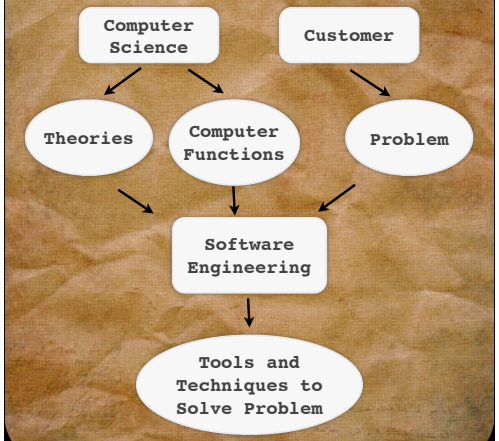
Software Engineering emerged as an effort to solve the “software crisis”.

In 2000, the Standish Group analysed software development projects

- 28% successfully completed
- 23% cancelled before completion
- 49 % over budget, late or with fewer functions than initially specified.



What is Software Engineering?



What is Software Engineering? -- One Possible Answer

Software engineering is an engineering discipline which is concerned with all aspects of software production

Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available

Software Engineering

Software engineering deals with the problems that arise when programs are:

- large
- involve many programmers
- exist over a long period of time

The nature of Software Engineering

Software Engineering resembles many different fields in many different ways:

- Mathematics
- Science
- Engineering
- Manufacturing
- Project management
- Art
- Performance

What are software engineering methods?

Structured approaches to software development which include system models, notations, rules, design advice and process guidance.

Model descriptions"

- Descriptions of graphical models which should be produced;

Rules

- Constraints applied to system models;

Recommendations

- Advice on good design practice;

Process guidance

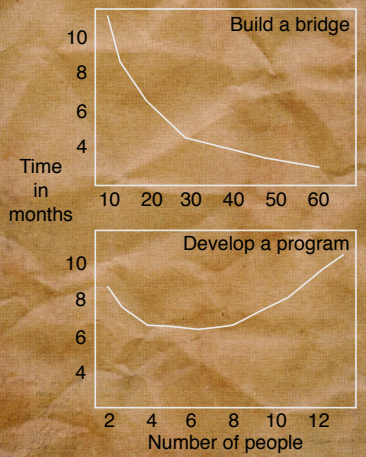
- What activities to follow.

Scope of Software Engineering (cont!d)

Why cannot bridge-building techniques be used to build operating systems?

- Attitude to collapse
- Imperfect engineering
- Complexity
- Maintenance

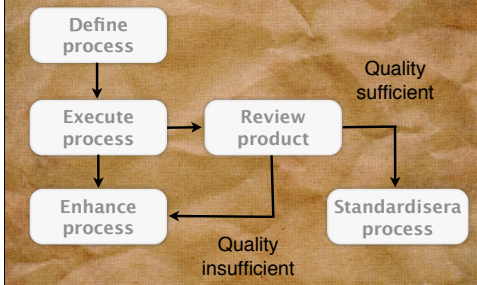
The Mythical Man-month



Cost to Correct Errors Found

Phase	Relative Time to Fix
Requirements	1
Specification	2
Design	4
Implementation	10
Integration	30
Maintenance	200

Manufacturing Process

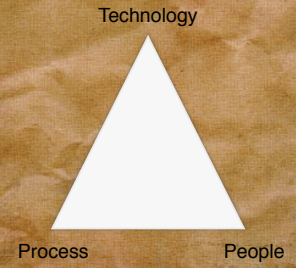


Which Factors Have Impact on Software Quality?

Technology used in development

Skills (and other things) of the people in the project

The quality of the software process



Improving the Process

Make sure everyone does things the same way

Define routines describing how different work tasks should be done

Define standards describing what the products of the work should look like

Collect metrics and experience

Identify strengths and weaknesses

Use available data and experience to improve

But

All “classical” software processes assume that software development is analogous to a defined industrial process that can be designed and run repeatedly with predictable results

- based on physical engineering processes
- predictive
- assume that people can be treated as resources

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.

K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas

Agile Misconceptions

When we speak of Agile development, listeners hear "agile" and interpret it as something fast-moving, something rapidly shifting gears and changing often

Certainly, many Agile software projects do change rapidly and move quickly, but not all of them and this is not their only characteristic

Agile Misconceptions, cont!d

People tend to distort the intended meaning of Agile, and that even includes some of the experts or people who have been involved in the Agile movement for some time.

Many have come to the conclusion that Agility is like art: "I know it when I see it," and "It's a very personal definition."

Practices can be applied to support the four values, but they are not, in themselves, Agile.

Agile Misconceptions, cont!d

Many people look a the Agile values and question the dichotomies embodied in them. For instance, is "comprehensive documentation" at the opposite pole from "working software"? The values seem to imply that these two aspects are opposed to each other, but in fact there's no logical justification for that

You cannot consider being Agile unless you agree to the pairing and choose to value the first value over the second in each pair

Agile Misconceptions, cont!d

- Is a silver bullet
- Will solve my resource issues
- Has no planning/ documentation/ architecture etc.
- Is a license to hack
- Creates quality issues
- Is undisciplined
- Doesn!t build on my previous experience/expertise
- Is not proven
- Is not being used by industry leaders

Agile principles

- 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.

Agile Risk Management

- Schedule slips -> Short release cycles
- Project cancelled -> Smallest release that makes sense
- System goes sour -> Maintain a suite of tests
- Defect rate -> Testing by programmers and customers
- Business misunderstood -> Make the customer part of the team
- Business changes -> Short release cycles
- False feature rich -> Address only the highest priority tasks

Agile principles seem simple enough...

... then why does it seem so difficult to implement them in practice?

Agile principles, forts.

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 maximising the amount of work not done -- is essential.

The best architectures, requirements, and designs emerge from self-organising teams.

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

What!s the Difference?

Approach built on adaptation rather than prediction

Focus on change rather than trying to prevent it from happening

This doesn!t mean that the work lacks neither structure or discipline

- Well-defined processes are followed, which makes a big difference between agile development ant “wild-west-hacking”

Examples of Agile Methods

eXtreme Programming (XP) -- Kent Beck, Ward Cunningham, Ron Jeffries

Scrum -- Jeff Sutherland and Ken Schwaber

Crystal Methods -- Alistair Cockburn

Feature Driven Development -- Jeff DeLuca

among others

AGILE Software Construction?

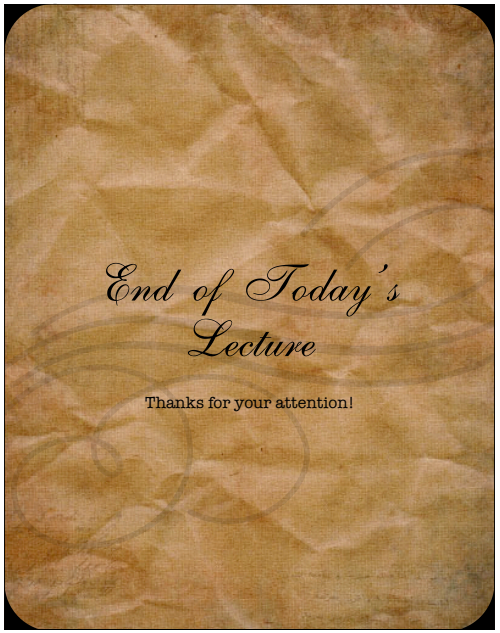
Agile methods is an addition to the tool-box of software constructors

Agile methods give us new strategies for software construction

Using agile methods, you still need to practice classical software engineering tasks like planning, requirements analysis, design, coding, testing, and documentation

Agile methods aim at further decreasing the risks involved in software development

Using agile methods does not mean that the work will be undisciplined


End of Today's Lecture
Thanks for your attention!