Software development processes
Life-cycle Models
Software Projects

Lecture 2

Kari Systä
For next week’s weekly exercise

• You either need user account to ”Lintula”
  – Beside student entrance to TC217
  – There are mailboxes for applications
• Or, bring your own laptop and
  – Install processing to you own laptop from [www.processing.org](http://www.processing.org)

• Week after you can also choose between PCs in TC217 and your own laptop
About the project / assignment

• Will be done in groups of 4
• Canvas for registering will be opened 19.1 and deadline is 26.1 (end of the day).
  – Groups fixed by 31.1
  – If you do not have complete group, you should still register. The staff will combine.
• Project will run in 4 Sprints

• Instructions (still under preparation)
Learning goals of today and the whole course

• What are process models and why they exist?
• Know basics of a few well-known process models
• And what are the motivations behind the models
  – To ”behave better”
    – (Some day) to select life-cycle model for your organization
• Know how to participate in the work efficiently
<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.01</td>
<td>Introduction to the course and software engineering</td>
</tr>
<tr>
<td>19.01</td>
<td>Life-cycle/process models</td>
</tr>
<tr>
<td></td>
<td>Project planning (part 1)</td>
</tr>
<tr>
<td>26.01</td>
<td>Scrum (part 1)</td>
</tr>
<tr>
<td>02.02</td>
<td>Requirement management</td>
</tr>
<tr>
<td>09.02</td>
<td>Version and configuration management</td>
</tr>
<tr>
<td>16.02</td>
<td>Scrum part 2. Roles of Scrum master, product owner. How to be use backlog as a RM tool.</td>
</tr>
<tr>
<td>23.02</td>
<td>Project planning (part 2): effort estimation</td>
</tr>
<tr>
<td>02.03</td>
<td>Modern practices Continuous Integration, Continuous Deployment, DevOps,..</td>
</tr>
<tr>
<td>09.03</td>
<td>No lectures - exam week</td>
</tr>
<tr>
<td>17.3</td>
<td>Improving Quality: review practices, testing and quality assurance</td>
</tr>
<tr>
<td></td>
<td>Quality in general; Quality management systems</td>
</tr>
<tr>
<td>23.3</td>
<td>Dependable and safety-critical systems</td>
</tr>
<tr>
<td>30.3</td>
<td>Other methods; RUP, XP, Lean; Kanban</td>
</tr>
<tr>
<td>37.3</td>
<td>Easter - no lectures</td>
</tr>
<tr>
<td>13.4</td>
<td>Role of software architecture; software maintenance; software</td>
</tr>
<tr>
<td></td>
<td>evolution</td>
</tr>
<tr>
<td>20.4</td>
<td>Software business, software start-ups</td>
</tr>
<tr>
<td>27.4</td>
<td>Guest lecture 1</td>
</tr>
<tr>
<td>4.5</td>
<td>Guest lecture 2</td>
</tr>
</tbody>
</table>
Life-cycle models


• A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed.

• Guideline to organize, plan, staff, budget, schedule and manage software project work over organizational time, space, and computing environments.

• Prescriptive outline for what documents to produce for delivery to client.

• Basis for determining what software engineering tools and methodologies will be most appropriate to support different life cycle activities.

• Framework for analyzing or estimating patterns of resource allocation and consumption during the software life cycle (Boehm 1981)

• Basis for conducting empirical studies to determine what affects software productivity, cost, and overall quality.
Software Process Models

• In contrast to software life cycle models, software process models often represent a networked sequence of activities, objects, transformations, and events that embody strategies for accomplishing software evolution.

• Such models can be used to develop more precise and formalized descriptions of software life cycle activities.
Life-cycle model and process model

• Often seen as synonyms. For example wikipedia.org/wiki/Software_development_process writes:
  – A software development process, also known as a software development life-cycle (SDLC), is a structure imposed on the development of a software product.

• https://ece.uwaterloo.ca/~se464/06ST/lecture/02_life-cycle-models.pdf:

• Lifecycle models: Phases in the life of an artifact, e.g., a system

• Process models: Activities performed on artifacts, e.g., development activities
Don’t PANIC

• We mostly do not care about the possible difference these terms in this course

• It is about rules and strategies for the “collaborative game”
Waterfall model - simplified
Royce, 1970

Principles of waterfall

• Waterfall is often understood as one-directional flow, but Royce considered iterations as a crucial part of the model. (mainly between consecutive steps)
• Waterfall is a plan-driven approach
• Move from step to next is a decision and often involves reviews, re-planing, budget decisions etc.
• Proper design and plan prevents extra work (and cost) in next steps
• The earlier the mistake is done, the more expensive it is
• Waterfall is consistent with other engineering processes
Documentation is a crucial part of waterfall
(one possible example)
Problems with waterfall

- Does not support division of the software to distinct stages
  - It is difficult to take out and use partial functionality
- Difficult to respond to changing customer requirements
- Management and motivation challenges of developers
  - Does not utilize full talent and motivation of talented and highly trained software developers
  - Does not show trust and empowerment

- Usually, waterfall is considered suitable for projects where
  - Requirements can be known in advance
  - Milestone reviews and audits are needed for example by security standards,
For example

Write the following words in alphabetical order
(the order they come in the alphabet)

apple  pumpkin  log  river  fox  pond

1. gelpp
2. ikmnppu
3. glo
4. eirry
5. fox
6. drop
Prototyping

• Motivations:
  – Get feedback
  – Ensure that selected technology works
  – Gain commitment

• Evolutionary prototype
  – Stepwise development towards product

• Throw-away prototype
  – Allows optimization

• Can be combined with waterfall
Precursor of interactive models: Spiral Model
(picture from: http://www.sei.cmu.edu/reports/00sr008.pdf)

Figure 1: Original Diagram of Spiral Development
Fundamentals of spiral model

- Spiral model is a risk-driven process and handling risks is explicit
- Each loop is split to four sectors
  1) Objective setting
  2) Risk assessment and reduction
  3) Development and validation
  4) Planning

- Spiral model is not a series of waterfalls! (A common misconception)
Rational Unified Process (RUP)

• Derived from UML and Unified Software Development Process
• Large and complex, but the purpose is not that all companies adopt all practices
• Three views
  – Dynamic: phases over time
  – Static: process activities
  – Practice

Next 2 slides kept as a reference. Not discussed in the lecture.
Phases in RUP

- **Inception**: business case and stakeholders
- **Elaboration**: spec, design, plan
- **Construction**: "the real work"
- **Transition**: to users
Iterative model: RUP
(Rational Unified Process)

Iterative Development
Business value is delivered incrementally in time-boxed cross-discipline iterations.

<table>
<thead>
<tr>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>E1</td>
<td>E2</td>
<td>C1</td>
</tr>
</tbody>
</table>

- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment

Time
Different projects - product

Vendor

specification ➔ Implem. ➔ validation ➔ ppackaging

research

Sales

Customer
Different project – customer specific

Vendor

research

bid

Implem.

validation

specification

deployment

Customer

Call for tender

specification

deployment

TIE-21100&21106/K.Systä
Different project – customer specific

Vendor

- research
- Specificat.
- Order spec

Customer

- Tender call
- bid

Vendor

- Implement.
- bid
- order
- validation
- deployment

TIE-21100&21106/K.Systä
Will that work?

• Assumption 1: good requirements can be written if enough effort is put on them
  – But: customer needs change over the time – even during the project
  – But: software is abstract until it is seen and tried
• Assumption 2: changes are small
  – But: they are not (and address surprising parts)
• Assumption 3: Integration is as easy as glue components together
  – But: the components are implemented by humans
• Assumption 4: schedule is followed
  – Actually very seldom
Iterative, agile

Vendor

- research
- bid

Customer

- Tender call

TIE-21100&21106/K.Systä

Demoable software
If possible, software can be taken into use.
Agile –manifesto

- February 2001
- 17 ”inventors”
- 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.
Manifesti Suomeksi

Me etsimme parempia keinoja ohjelmistojen kehittämiseen tekemällä sitä itse ja auttamalla siinä muita. Tässä työssämme olemme päätyneet arvostamaan

• **Yksilöitä ja vuorovaikutusta** enemmän kuin prosesseja ja työkaluja

• **Toimivaa sovellusta** enemmän kuin kokonaisvaltaista dokumentaatiota

• **Asiakasyhteistyötä** enemmän kuin sopimusneuvotteluita

• **Muutokseen reagoimista** enemmän kuin suunnitelman noudattamista.

Vaikka oikeallakin puolella on arvoa, me arvostamme vasemmalla olevia asioita enemmän.

19.1.2015
## Five principles of Agile

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer involvement</td>
<td>Through the project. Provide and prioritize requirements, evaluate iterations</td>
</tr>
<tr>
<td>Incremental delivery</td>
<td>Customer specifies the increments</td>
</tr>
<tr>
<td>People not process</td>
<td>Skill recognized and exploited; Team should decide on ways of working</td>
</tr>
<tr>
<td>Embrace change</td>
<td>Plan and design for change</td>
</tr>
<tr>
<td>Maintain simplicity</td>
<td>Both in process and software</td>
</tr>
</tbody>
</table>
Plan-driven vs. agile specification

Requirements engineering → Requirements specification → Design and implementation

Requirements engineering ← Design and implementation
Problems in realizing Agile

• Getting customer commitment and trust is difficult
• Some team members do not have suitable personalities
• Prioritizing changes is difficult – especially if there are many stakeholders
• Maintaining simplicity requires extra work
• Cultural changes through the company
  (Agile training should start from management)
Extreme programming (XP)

• Well-known method developed by Kent Beck
• (only briefly covered by Haikala&Mikkonen, but internet is full of resources)

• Requirements are expressed as scenarios called *User Stories* which are implemented directly as series of tasks
• Programmers work in pairs (Pair Programming) and develop tests for each task before writing the code (Test-driven development)
• All tests must be successfully executed when new code is integrated
XP (Extreme programming)

http://www.extremeprogramming.org/
<table>
<thead>
<tr>
<th>Practice/principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental planning</td>
<td>Minimun useful is implemented first; frequent releases</td>
</tr>
<tr>
<td>Small releases</td>
<td>Minimun useful is implemented first; frequent releases</td>
</tr>
<tr>
<td>Simple design</td>
<td>Spend enough time in design</td>
</tr>
<tr>
<td>Test-driven development</td>
<td>Test written before code, automated tests</td>
</tr>
<tr>
<td>Refactoring</td>
<td>All team members should refactor code to keep simple and maintainable</td>
</tr>
<tr>
<td>Pair programming</td>
<td>Check each others’ work; support</td>
</tr>
<tr>
<td>Collective ownership</td>
<td>No islands of responsibilities; every body can change everything</td>
</tr>
<tr>
<td>Continuous integration</td>
<td>Whenever something is ready it is integrated; always test</td>
</tr>
<tr>
<td>Sustainable pace</td>
<td>Large amounts of overtime is not sustainable</td>
</tr>
<tr>
<td>On-site customer</td>
<td>Continuous access to customer</td>
</tr>
</tbody>
</table>
About pair-programming

• Supports collective ownership and responsibility
• Informal review because each line of code has been seen more than one person
• Supports refactoring
• Fosters learning from colleagues
• Research on productivity gives mixed results
Scrum

• Framework for agile and iterative development
• Sutherland, Jeffrey Victor; Schwaber, Ken “Business object design and implementation” OOPSLA '95 workshop proceedings.
XP vs Scrum

• XP has typically shorter iterations (1-2w instead of 2-4w)
• Scrum does not allow changes into sprints
• XP is work in strict priority order
• Scrum does not prescribe any engineering practices
• Scrum focuses more on management aspects
Learning goals of today and the whole course

• What are process models and why they exist?
• Know basics of a few well-known process models
• And what are the motivations behind the models
  – To ”behave better”
  – (Some day) to select life-cycle model for your organization
• Know how to participate in the work efficiently
Project

• Wikipedia (borrows): In project management a project consists of a temporary endeavor undertaken to create a unique product, service or result.

• Oxford dictionary an individual or collaborative enterprise that is carefully planned and designed to achieve a particular aim.
Key elements for this lecture

- Wikipedia (borrows): In project management a project consists of a temporary endeavor undertaken to create a unique product, service or result.
- Oxford dictionary an individual or collaborative enterprise that is carefully planned and designed to achieve a particular aim.
A project

Has
• Start
• End
• Goals

Should have
• Plan
• Success criteria
• Management
Project management aims at:

• Deliver SW to customer at the agreed time
• Keep overall costs within the budget
• Deliver software that meets customer’s expectations
• Maintain spirit and performance of the team
Special about software projects

• The product is abstract and thus intangible
  – Unlike buildings it is hard to tell condition and phase of development

• Large software project are often one-off projects
  – Hard to repeat experience

• Processes are variable and organization specific
Project organization (traditional)

Customer

Steering group

Vendor

Support group
- Tech experts
- Legal experts

Other stakeholders

Project manager of customer

Users

PM of vendor
Project planning

• Organization
• Target setting
• Risk analysis
• Selection of technologies, methods, practices, tools
  – For instance if agile is selected
• Support (documentation, quality assurance, product management)
• Splitting and phasing
  (WBS = Work Breakdown Structure)
• Worktime estimation
• Resource availability and time table
  – Developers, external experts
  – Special tools
• Budget, financing, funding
• Success criteria
Gantt – charts
(lähde: http://www.matchware.com)
Another alternative
(Lähde: http://orgmode.org)
Risk management

- Identification
  - Potential risks
- Analysis
  - Prioritized risk list
- Planning/
  - Mitigation
- Monitoring
  - Avoidance Contingency Mitigation
A very common approach for analysis and planning

- Probability of the risk
- Severity of the risk

⇒ Priority

Plan

- Ways to avoid the risk
- Ways reduce probability of the risk
- What do if the risk is realized (action)
- How we minimize the consequences in advance
## Simple table

<table>
<thead>
<tr>
<th>RISK</th>
<th>Probab.</th>
<th>Severity</th>
<th>Prevent</th>
<th>Reduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key person leaves</td>
<td>50%</td>
<td>8</td>
<td>Keep people happy. Give a raise.</td>
<td>Ensure that all critical data and skill is possessed by more than one.</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Nuclear war</td>
<td>&lt;0.001%</td>
<td>10</td>
<td>Attend peace campaign</td>
<td>Move the office to a bunger</td>
</tr>
</tbody>
</table>

Often better to use
- low
- medium
- high

27.1.2014
Reasons for one big project to fail

• The size and features of the result were impossible to estimate during planning
• The result was much bigger than originally planned
• There was a constant flow of changes: features were added and removed
• Coordination between sub-projects was mostly non-existing
• Terrible hurry during the last phases of the project
More reasons

• Close to end project manager got sick and replaced by unexperienced person
• Many features and technologies were new and never tested in similar products.
• System testing showed that the system is not stable, but because project was late it was deployed anyways.
more

• Afterwards it was discovered that the design could not work. The ship would have been stable only with so much of weight that some guns had been under water surface.
  • Reference; Curt Borgenstam: Why the Wasa Capsized.
Why project plan is important

• It is a tool for steering and tracking
• Communication tool among the stakeholders
  – Goals
  – Tasks and timetable
  – Organization and responsibilities
  – Tools, working practices
  – Budgets
  – Risks
A few words about project management and agile
## Five principles of Agile

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer involvement</td>
<td>Through the project. Provide and prioritize requirements, evaluate iterations</td>
</tr>
<tr>
<td>Incremental delivery</td>
<td>Customer specifies the increments</td>
</tr>
<tr>
<td>People not process</td>
<td>Skill recognized and exploited; Team should decide on ways of working</td>
</tr>
<tr>
<td>Embrace change</td>
<td>Plan and design for change</td>
</tr>
<tr>
<td>Maintain simplicity</td>
<td>Both in process and software</td>
</tr>
</tbody>
</table>
But...

• Some things are not covered by agile methods
  – Stakeholders
  – Budget
  – Risks
  – High-level goals
  – Risks

• Some things need to be agreed on
  – Timing of sprints
  – Who participate in sprint review
  – Who takes the roles
    (in Scrum: Scrum master and product owner)
Agile improves visibility to management

• Important argument for selling the idea of Agile
• Each sprint provides an opportunity to see the progress
  – Steering group or management may ask a status report
    (but status reporting and sprint rhythm should be synchronized)

• Example: burn-down charts