Agenda:
Definition of process
Life-cycle models
  V and Waterfall
  Incremental and Iterative
Method frameworks
  OpenUP
  Essence Kernel
  eXtreme Programming
  SCRUM
  KANBAN
Requirements

System Design (Architecture, High-level Design)

Module Design (Program Design, Detailed Design)

Implementation of Units (classes, procedures, functions)

Unit testing

Validation of Requirements, Verify Specification

Verify System Design

Verify Module Design

Verify Implementation

Maintenance

Acceptance Test (Release testing)

System Testing (Integration testing of modules)

Module Testing (Integration testing of units)
Remember the necessary parts of a project?

- An order
- A budget
- A single item occurrence
- Time
- Goal
Processes are reoccurring

- Ordered set of activities
- May contain sub-processes
- Goal of each activity
- Each activity has entry/exit criteria and input/output
- Constraints
Agenda:
- Definition of process
- Life-cycle models
  - V and Waterfall
  - Incremental and Iterative
- Method frameworks
  - OpenUP
  - Essence Kernel
  - eXtreme Programming
  - SCRUM
  - KANBAN
Remember our life-cycle model?...
... also known as the V-model

- Requirements
  - Validate Requirements, Verify Specification
  - Acceptance Test (Release testing)
- System Design (Architecture, High-level Design)
  - Verify System Design
  - System Testing (Integration testing of modules)
- Module Design (Program Design, Detailed Design)
  - Verify Module Design
  - Module Testing (Integration testing of units)
- Implementation of Units (classes, procedures, functions)
  - Verify Implementation
  - Unit testing

Time
Now, remove the abstraction level ...

- **Requirements**
  - Validate Requirements, Verify Specification
  - Acceptance Test (Release testing)

- **System Design** (Architecture, High-level Design)
  - System Testing (Integration testing of modules)

- **Module Design** (Program Design, Detailed Design)
  - Module Testing (Integration testing of units)

- **Implementation** of Units (classes, procedures, functions)
  - Unit testing

- **Time**

- **Maintenance**
... and we got the waterfall model!

- One of the first life-cycle models (Royce, 1970)
- The waterfall development model originates in the manufacturing and construction industries
- Very common, very criticized
The classical waterfall model

1. Requirements
2. System Design
3. Module Design
4. Implementation
5. Unit Testing
6. Module Testing
7. System Testing
8. Acceptance Test
9. Maintenance

Milestone and deliverable at each step. (Artifacts such as Design document, Req. Specification, etc.).

Finish each phase before continue to next.

Processes and Life Cycles/K Sandahl

2021-09-21

11
Problems with the waterfall model

- Software requirements change, hard to sign-off on a SRS.
- Early commitment. Changes at the end, large impact.
- Feedback is needed to understand a phase. E.g. implementation is needed to understand some design.
- Difficult to estimate time and cost for the phases.
- Handling risks is not an explicit part of the model. Pushes the risks forward.
- Software "is not" developed in such a way. It evolves when problems are more understood. Little room for problem solving.
Advantages with the waterfall model

• Simple, manageable and easy to understand
• Fits to common project management practices (milestones, deliverables etc.)
• Can be suitable for short projects (some weeks)
• Can be used at a large system level (several years)
• Can be suitable for "stable" projects, where requirements do not change
• Focus on documents, saves knowledge which can be reused by other people.
• Can be suitable for fixed-price contracts
Do it twice (Royce, 1970)

First round, a prototype

Second round, do it right.

Input to the phases in the second round
Agenda:
Definition of process
Life-cycle models
  V and Waterfall
  Incremental and Iterative
Method frameworks
  OpenUP
  Essence Kernel
  eXtreme Programming
  SCRUM
  KANBAN
Iterative and Incremental methods

Sources: Activity village and Lego
Iterative development

*When should the releases take place?*

**Time-boxing** - The time period is fixed for each iteration.

*What should be included in the release?*

**Prioritized functionality** - Do the most important parts first.

Customer Feedback

---

**R1**

Iteration 1

---

**R2**

Iteration 2

---

**Final Release!**

Iteration 3

---

**Time**
Dependent project parameters revisited

Calendar time and resources are fixed

- Calendar Time
- Resources
- Features
- Quality

Select the most important functions
Select quality. E.g. how general should we be?
Prioritization of requirements

Customer Value

High

Low

Sweet Spot

Avoid

Development Effort
Problems with iterative development

- Problem with current business contracts, especially fixed-price contracts.
- With short iterations it can be hard to map customer requirements to iterations.
- Overhead added
- Requirements selection problem
- Stressful learning period if moving from the classical waterfall model
Advantages with iterative development

- Misunderstandings and inconsistency are made clear early (e.g. between requirement, design, and implementation)
- Encourage to use feedback -> elicit the real requirements
- Forced to focus on the most critical issues
- Continuous testing offers project assessment
- Workload is spread out over time (especially test)
- The team can get "lesson learned" and continuously improve the process
- Stakeholders get concrete evidence of progress
Agenda:
Definition of process
Life-cycle models
  V and Waterfall
  Incremental and Iterative
Method frameworks
  OpenUP
  Essence Kernel
eXtreme Programming
  SCRUM
  KANBAN
Processes, models, methodologies...

Process Models
- Waterfall model
- V-model
- Spiral model
- Prototype model

"what" at a high level of abstraction

Which is the "best" approach?

Method frameworks
- Extreme Programming (XP)
- Essence Kernel
- Open/UP
- Scrum
- Kanban
- Agile
- Lean

"what" and to a certain level "how"
Open/UP mimics the way software is developed

- Down-scaled variant of RUP
- Mapping
  - Roles
  - Tasks
  - Workproducts

The architecture notebook
Agenda:
Definition of process
Life-cycle models
  V and Waterfall
  Incremental and Iterative
Method frameworks
  OpenUP
  Essence Kernel
eXtreme Programming
SCRUM
KANBAN
Essence Kernel monitors the common denominator of all SE projects

Source of picture: Ivar Jacobsson International
Requirements – states

The need for a new system has been agreed.

The purpose and theme of the new system are clear.

The requirements provide a coherent description of the essential characteristics of the new system.

The requirements describe a system that is acceptable to the stakeholders.

Enough of the requirements have been addressed to satisfy the need for a new system in a way that is acceptable to the stakeholders.

The requirements have been addressed to fully satisfy the need for a new system.
Snap-shot of relations between elements
Agenda:

Definition of process

Life-cycle models
  V and Waterfall
  Incremental and Iterative

Method frameworks
  OpenUP
  Essence Kernel
  eXtreme Programming
  SCRUM
  KANBAN
Agile Approaches - Agile Alliance

Lightweight approaches to satisfy the customers with "early and continuous delivery of valuable software"

Manifesto for Agile Software Development

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

(http://agilemanifesto.org, 2001)
Extreme Programming

- Formulated in 1999 by Kent Beck
- XP is “a light-weight methodology for small to medium-sized teams developing software in the face of vague or rapidly changing requirements.”
- Driving good habits to the extreme
XP Values

• Communication
  – On-site customer, user stories, pair programming, daily standup meetings, etc.

• Simplicity
  – "Do the simplest thing that could possibly work" (DTSTTTPCW) principle

• Feedback
  – Unit tests tell programmers status of the system
  – Programmers produce new releases every 2-3 weeks for customers to review

• Courage
  – Communicate and accept feedback, throw code away, refactor the architecture of a system
Pair Programming
- Programming as a collaborative conversation
- Focus on task
- Clarify ideas
- Rotate frequently

Refactoring
- Improve the design of existing code without changing its functionality
- Tool support, e.g. Eclipse

Continuous Integration
- Integrate and test often
- Automated build system
- Automated regression tests (e.g. JUnit)

Stories
- "requirements", but not mandatory
- A token for a piece of system capability to be implemented
- Name + short story
- On index cards (paper)

Test-First Programming
- Create tests before code
- Focus on interface and "what is needed"
- Gets tests for free
Agenda:
Definition of process
Life-cycle models
V and Waterfall
Incremental and Iterative
Method frameworks
OpenUP
Essence Kernel
eXtreme Programming
SCRUM
KANBAN
Scrum

Approach public in 1995 at OOPSLA

"Scrum" strategy used in rugby for getting an out-of-play ball back into play.
Scrum in a nutshell

Small, cross-functional **teams**

**Product** split into small, roughly estimated, stories

**Iterations** - sprints

Continuous **improvement and deployment**

Slides by Aseel Berglund
The Sprint

Sprint end date and deliverable do not change
The Team

Sprint end date and deliverable do not change
The Scrum Master

Scrum Master

The Team

Sprint end date and deliverable do not change
The Product Owner

Inputs from Executives, Stakeholders, Customers, Users, Team

Scrum Master

The Team

Sprint end date and deliverable do not change
The Product Backlog

Inputs from Executives, Stakeholders, Customers, Users, Team

Scrum Master

Product Owner

The Team

A prioritized list of what is required, features, stories

Product Backlog

Sprint end date and deliverable do not change
The Sprint Planning Meeting

Inputs from Executives, Stakeholders, Customers, Users, Team

Scrum Master

Product Owner

The Team

Product Backlog

A prioritized list of what is required, features, stories

1_2_3

Team selects starting at top as much as it can commit to deliver by end of sprint

Sprint Planning Meeting

Sprint end date and deliverable do not change
The Sprint Backlog

Inputs from Executives, Stakeholders, Customers, Users, Team

Scrum Master

Product Owner

The Team

Sprint Planning Meeting

Sprint Backlog

Task Breakout

Sprint end date and deliverable do not change

A prioritized list of what is required, features, stories

Team selects starting at top as much as it can commit to deliver by end of sprint
Sample Taskboard
The Daily Scrum Meeting

Inputs from Executives, Stakeholders, Customers, Users, Team

Scrum Master

Product Owner

The Team

Daily Scrum Meeting

1. Team selects starting at top as much as it can commit to deliver by end of sprint
2. A prioritized list of what is required, features, stories

Sprint Planning Meeting

Sprint Backlog

Task Breakout

Sprint end date and deliverable do not change

1-4 Week Sprint

Every 24 Hours

Processes and Life-Cycles/K Sandahl 2021-09-21
The Burn Down Charts

Inputs from Executives, Stakeholders, Customers, Users, Team

Scrum Master

The Team

Product Owner

Product Backlog

2021-09-21

Processes and Life-Cycles/K Sandahl

Daily Scrum Meeting

Burn down charts

Every 24 Hours

1-4 Week Sprint

Sprint end date and deliverable do not change

Team selects starting at top as much as it can commit to deliver by end of sprint

Task Breakout

Sprint Backlog

Sprint Planning Meeting
The burn down chart

- Only track hours remaining, not hours worked
- X – days (in Sprint)
- Y – hours remaining in estimated time or points
- Remove meeting time, vacation etc. from total available hours
- Update only when PBIs are DONE

- When not done – Undone PBIs
The Sprint Review Meeting

Inputs from Executives, Stakeholders, Customers, Users, Team

Scrum Master

Product Owner

The Team

Task Breakout

Sprint Planning Meeting

Sprint Backlog

Burn down charts

Every 24 Hours

1-4 Week Sprint

Sprint end date and deliverable do not change

2021-09-21

Finshed work

Done?

A prioritized list of what is required, features, stories

Team selects starting at top as much as it can commit to deliver by end of sprint

Product Backlog

Inputs from Executives, Stakeholders, Customers, Users, Team

Scrum Master

Burn down charts

Every 24 Hours

1-4 Week Sprint

Sprint end date and deliverable do not change

2021-09-21

Finshed work

Done?

A prioritized list of what is required, features, stories

Team selects starting at top as much as it can commit to deliver by end of sprint

Product Backlog
The Definition of Done!

- When are we done?
- “No more remaining work”
- Includes testing, documentation etc.
- Possible to ship after each sprint
- Everybody – understand what done means

Tools to support done
- Version handling (SCM)
- Automated build
- Automated tests (Continuous integration)
The Sprint Retrospective

Inputs from Executives, Stakeholders, Customers, Users, Team

Scrum Master

Product Owner

The Team

Product Backlog

A prioritized list of what is required, features, stories

Team selects starting at top as much as it can commit to deliver by end of sprint

Sprint Planning Meeting

Task Breakout

Sprint Backlog

Sprint end date and deliverable do not change

Burn down charts

Every 24 Hours

1-4 Week Sprint

Sprint review meeting

Finished work

Sprint retrospective

Daily Scrum Meeting
Agenda:
Definition of process
Life-cycle models
  V and Waterfall
  Incremental and Iterative
Method frameworks
  OpenUP
  Essence Kernel
  eXtreme Programming
  SCRUM
  KANBAN
Lean methods according to Masayuki Yamaguchi(*)

"West"

Japan

(*) Mary and Tom Poppendieck: Leading lean software development Addison-Wesley 2010
Lean principles

- Eliminate waste – don’t develop the wrong product
- Build quality in – automate tedious or error prone parts
- Create knowledge – continuous process improvement
- Defer commitment – wait until facts are known
- Deliver fast – limit queues
- Respect people – self-organized teams
- Optimize the whole – don’t just fix bugs, solve problems

(*) Mary and Tom Poppendieck: Leading lean software development Addison-Wesley 2010
Kanban

The two pillars of the Toyota production system are just-in-time production and automation with a human touch, or autonomation. The tool used to operate the system is kanban.
看板 - Kanban

- Kanban is a Japanese word that means “visual card,” “signboard,” or “billboard.”
- Toyota originally used Kanban cards to limit the amount of inventory tied up in “work in progress” on a manufacturing floor.
- Kanban is a **lean** approach to agile software development.
- Focuses on the flow of progress.
Which strategy do you prefer?

You have three books to read before the exam. Do you

1. Work "little and often", you study each course for three hours per day; or

2. "Reduce multitasking", read the first book, then the second, and then the third?

https://www.menti.com/m5eyksempa
How does Kanban Work?

• **Visualize the workflow**
  – Split the work into pieces, write each item on a card and put on the wall.
  – Use named columns to illustrate where each item is in the workflow.

• **Limit WIP** (work in progress) – assign explicit limits to how many items may be in progress at each workflow state.

• **Measure the lead time** (average time to complete one item, sometimes called “cycle time”), optimize the process to make lead time as small and predictable as possible.
A simple Kanban Board

Source: http://www.crisp.se/gratis-material-och-guider/kanban
Work In Progress

Work In Progress, WIP, limits are designed to:

- reduce multitasking
- maximize throughput
- enhance teamwork

Reducing multitasking is beneficial for two primary reasons
Reducing Multitasking

20% time is lost to context switching per ‘task’, so fewer tasks means less time lost

(from Gerald Weinberg, Quality Software Management: Systems Thinking)
Reducing Multitasking

Performing tasks sequentially yields results sooner.

Multi-tasking A, B and C (on the top), delivers A much later, and even C slightly later, than sequentially (on the bottom).
Typical Measurements

- **Cycle time** – Measured from when you started working on it
- **Lead time** – Measured form when the customer ordered
- **Quality** – Time spent fixing bugs per iteration
- **WIP** – Average number of “stories” in progress
- **Throughput** – Number of “stories” completed per iteration (when using fixed iterations)
Benefits of Kanban

• Eliminate over-production, the #1 waste
• Produce only what is ordered, when ordered, & quantity ordered
• Increase flexibility to meet customer demand
• Competitive advantage by sequencing shipments to customers (what they want, when they want it, in the order they want it!)
• Several things are optional: sprints, estimation, agile practices. Even iterations!