SYSTEMS PLANNING

Software Process Models

Class content

What to expect from this class?

- An **introduction to general process models**
- A description of the **software specification phase**
- A description of the **design and implementation phase**
- A description of the **validation phase**
- A description of the **evolution phase**
- Integration of **change**
- Introduction to the **Rational Unified Process**

Source: https://pixabay.com/en/checklist-check-list-marker-2077020/
by TeroVesalainen |CCO Creative Commons

Class learning goals

MCI MANAGEMENT CENTER INNSBRUCK

What should be your learning outcome?

- Understand the different phases of the **software process**
- Know **general process models** and their advantages and challenges
- Know how formal process models may handle **change**
- Understand the **principles** of the **Rational Unified Process**

Source: https://pixabay.com/en/learning-hint-school-subject-3245792/ by harishs |CCO Creative Commons

The Software Process

MCI MANAGEMENT CENTER INNSBRUCK

The Software Process

Leads to the development of a software product

- **Software specification**
  The **functionality** of the software and constraints on its operation must be defined
- **Software design and implementation**
  The software to **meet the specification** must be produced
- **Software validation**
  The software must be **validated** to ensure that is does what the customer wants
- **Software evolution**
  The software must **evolve** to meet **changing customer needs**

Source: https://pixabay.com/en/ux-design-webdesign-app-mobile-787980/ by FirmBee |CCO Creative Commons

The Software Process

These are the overall activities but there are also

- **Sub-activities**
  e.g. requirements validation, architectural design unit testing, etc.
- **Support process activities**
  e.g. documentation, software configuration management

Source: https://pixabay.com/en/idea-empty-paper-pen-light-bulb-1876658/ by qimono |CC0 Creative Commons

The Software Process

The Software Process

- **Software processes are complex**
  They rely on people making **decisions and judgments**
- **There is no ideal process**
  organizations come up with their **own process;** they evolve around the people and organization involved;

Source: https://pixabay.com/en/time-money-horizontal-speculate-1019889/ by 3dman_eu |CC0 Creative Commons

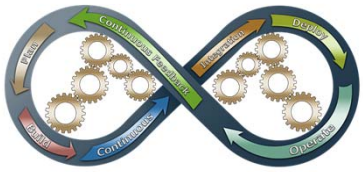
However:

- **Critical systems** usually need a very **structured** process
- **Business systems** may be developed applying a **less formal** approach (rapidly changing requirements)

The Software Process

Categories of software processes

- **Plan-driven processes**
  Activities are planned in advance and progress is measured against this plan
- **Agile processes**
  Planning is incremental;
  Easier to reflect changing customer requirements



Source: https://pixabay.com/en/devops-business-process-improvement-3148393/
by diwou | CCO Creative Commons

Possible improvements in both cases

- Process standardization (leads to)
- Improved communication

Generic Process Models

(sometimes called 'process paradigms')

- **The Waterfall Model**
  Fundamental process activities as separate process phases i.e. Specification, design, implementation, testing, etc.
- **Incremental Development**
  Interleaves activities;
  Develops a series of versions always adding functionality;
- **Reuse-oriented Software Engineering**
  Based on already existing components;
  Focuses on component integration

Source: https://pixabay.com/en/machine-mechanical-eye-blue-look-1776925/
by intographics | CCO Creative Commons

Note: Often these different models are combined e.g. waterfall for specified components and incremental for the user interfaces.

Generic Software Process Models

The Waterfall Model

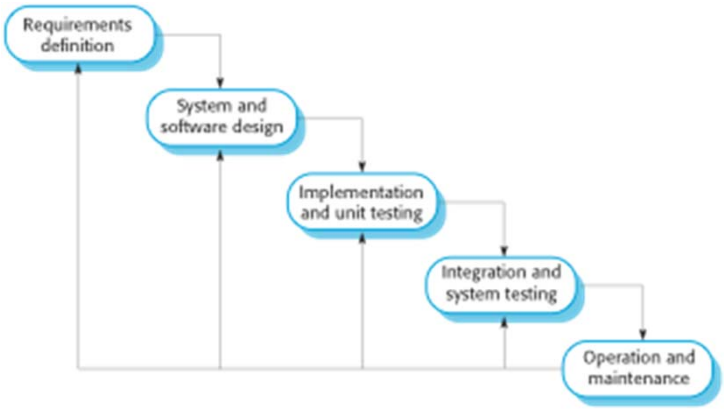
(based on system engineering; Royes, 1970)

- **Requirements analysis and definition**
  Constraints and goals are established with system users;
  Detailed system specification created;
- **System and software design**
  Allocation of requirements to hard-/software
  Description of the fundamental system
- **Implementing and unit testing**
  Program and test
- **Integration and system testing**
  Integrate with other components and test
- **Operation and maintenance**
  Install system and put into practice;
  Correct errors and improve implementation;

Source: https://pixabay.com/en/godafoss-iceland-waterfall-falls-1840758/ by 12019 |CCO Creative Commons

Generic Software Process Models

The Waterfall Model



Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

The Waterfall Model

- Result of each phase is one or more **documents** that need to be **approved** (i.e. signed off)
- **Stages overlap** and feed information to each other
- **Involves feedback** from one phase to the other
- Iterations can be **costly** because they need **documentation and improvement**
- This often **leads to pre-mature freezing of requirements**
- **Process is visible to mangers** so they can monitor progress

Source: https://pixabay.com/en/files-paper-office-paperwork-stack-1614223/ by myrfa |CC0 Creative Commons

- **Main problem:** Inflexible partitioning of the project into distinct stages, which leads to early commitments
- **Hence:** The waterfall model should **only be used with requirements that are well understood and unlikely to change!**
- **Yet, it is still very common!**

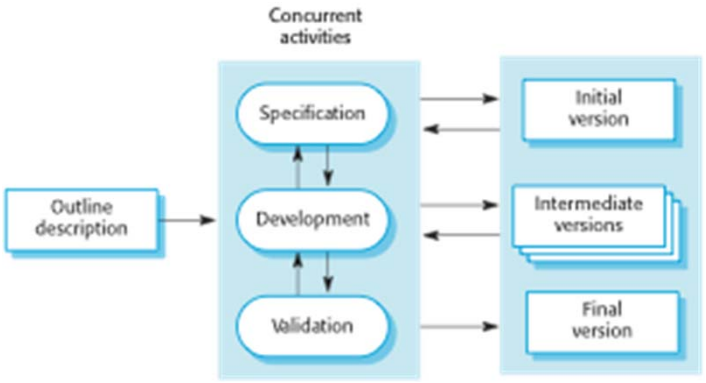
Incremental Development

- Initial implementation is **exposed to user comment**
- **Several versions** until the system has been developed
- **Rapid feedback** across activities
- Fundamental approach for **agile development**
- Better than waterfall model for most business, e-commerce, and personal systems
- **Cheaper and easier** to make changes
- Most urgently **required functionality is implemented first**

Source: https://pixabay.com/en/feedback-conversation-board-chalk-1186347/ by YvonneScholz |CC0 Creative Commons

Generic Software Process Models

Incremental Development



Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

Generic Software Process Models

Incremental Development

Important Benefits compared to waterfall model

- **Reduced cost of accommodating changing requirements**
  Less documentation than in waterfall model
- **Easier to get customer feedback**
  Demonstrators and prototypes are easier to evaluate
- **More rapid delivery and deployment of features**
  Most urgent features come first

Source: https://pixabay.com/en/like-facebook-social-media-icon-2195024/ by raphaelsilva |CC0 Creative Commons

Note: Incremental development is now the most common approach. It can be applied plan-driven, agile, or most commonly in a mixture of these approaches.

Generic Software Process Models

Incremental Development

Problems compared to waterfall model

- **Process is not visible to mangers**
  Regular deliverables are needed to measure progress;
  Less documentation is produced;
- **System structure tends to degrade over time**
  Refactoring should improve software quality;

Source: https://pixabay.com/en/boxes-cardboard-carrying-overload-2624231/ by skeeze | CC0 Creative Commons

Particularly **large, complex, long-lifetime systems** require a stable framework or architecture. This has to be planned in advance.

Generic Software Process Models

Reuse-oriented Software Engineering

- Majority of software projects **reuse software components**
- Often happens **informally**
- Components are **modified and incorporated**
- In the **21st century software development processes** based on reuse have become widely used
- These approaches **rely on a large base of reusable software components**

Source: https://pixabay.com/en/recycling-characters-waste-garbage-1341372/ by Elionas | CC0 Creative Commons

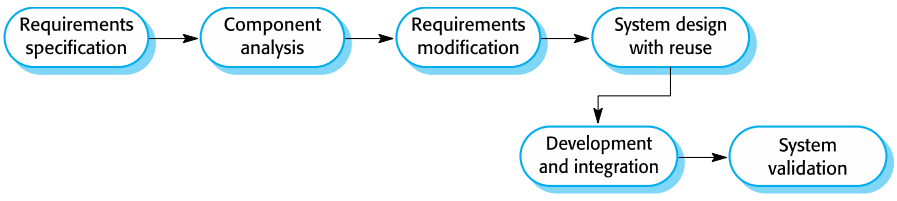
Reuse-oriented Software Engineering

Model stages

- **Components analysis**
  Search components that implement specification
- **Requirements modification**
  Adapt and modify components
- **System design with reuse**
  Design a framework that integrates components
- **Development and integration**
  Developing of missing components and integration work

Source: https://pixabay.com/en/steps-staircase-climbing-1081909/ by Free-Photos |CCO Creative Commons

Reuse-oriented Software Engineering



Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

Generic Software Process Models

Reuse-oriented Software Engineering

Three types of reusable software components

- **Web services**
  Developed according to service standards
- **Collections**
  Developed as packages (e.g. .Net, Java EE, etc.)
- **Stand-alone software**
  Configured in a particular environment

Source: https://pixabay.com/en/monitor-computer-screen-hardware-862116/
by geralt |CC0 Creative Commons

Note: Reuse-oriented SE reduces the amount of software to be developed and consequently reduces cost and risk. However, this also often leads to compromises.

Process Activities

Process activities

Software processes are interleaved sequences of

- Specification
- Design and Implementation
- Validation and
- Evolution

In the **waterfall** model they are organized in **sequence**

In the **incremental development** model they are **interleaved**

How they are carried out depends on

- Software to be developed
- People involved
- Organizational structures

Source: https://pixabay.com/en/analysis-automation-business-man-3606761/
by Mohamed_hassan |CC0 Creative Commons

Process Activities

MCI MANAGEMENT CENTER INNSBRUCK

Software Specification

- **Feasibility study**
  Is it feasible to build the required system?
  Is it cost-effective?
- **Requirements elicitation and analysis**
  Understand the user needs;
- **Product Brief (Lastenheft)**
  Translate user requirements into a formal document; include user requirements and system requirements;
- **Requirements validation**
  Check whether requirements are real, consistent and complete;
- **Requirements Specification Document (Pflichtenheft)**
  Describes how the requirements specification will be implemented;

Source: https://pixabay.com/en/triangle-quality-time-cost-3125882/ by diwou | CC0 Creative Commons

Process Activities

MCI MANAGEMENT CENTER INNSBRUCK

Software Design and Implementation

- **Converting a specification into an executable system**
- Involves **software design** and **programming** as well as **refinement**

Software design includes
- Description of the structure
- Data models
- Interfaces between system components
- Algorithms

Source: https://pixabay.com/en/programming-html-css-javascript-1873854/ by Boskampi | CC0 Creative Commons

It treats the **'software platform'** as an environment that executes and interfaces with other software. Different types of software require **different design activities**.

Process Activities

MCI MANAGEMENT CENTER INNSBRUCK

Software Design and Implementation

- **Architectural design**
  Overall structure of the system;
  Principle components and their relationships;
- **Interface design**
  Interface between components;
- **Component design**
  Functionality of components;
- **Database design**
  Data structures and the way they are represented in the database

Source: https://pixabay.com/en/white-board-startup-start-up-593300/ by StartupStockPhotos |CC0 Creative Commons

Process Activities

MCI MANAGEMENT CENTER INNSBRUCK

Software Validation

Verify that the system conforms with **specification** and **user expectations**. To do so, follow a three-stage testing process:

Source: https://pixabay.com/en/board-school-uni-learn-work-test-361516/ by geralt |CC0 Creative Commons

- **Development testing**
  Each component is tested independently (e.g. classes, functions, etc.);
  e.g. JUnit tests (Massol & Husted, 2003);
- **System testing**
  Test an integrated system;
  Check for unanticipated interactions between components;
- **Acceptance testing**
  Use real data as opposed to simulated test data;
  May also reveal additional problems where the system does not meet the user's needs;

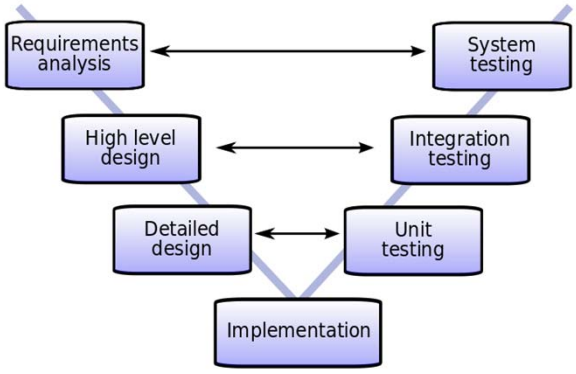
Software Validation

- Component development and testing are **interleaved**
- In extreme programming tests are **developed before development starts**
- Plan-driven software processes use test plans i.e. **the V-model of development**

Note:
- Acceptance testing with a single client is often called **alpha testing**
- Acceptance testing with a number of potential clients is called **beta testing**

Source: https://pixabay.com/en/checklist-clipboard-pen-paper-1643781/
by Deedster | CC0 Creative Commons

Software Validation – the V Model (Verification and Validation)

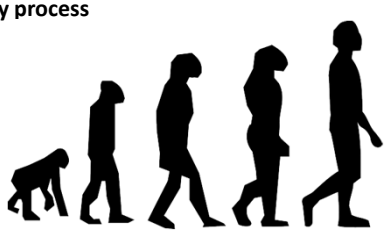


Source: https://commons.wikimedia.org/wiki/File:V-model.svg
by Herman Bruyninckx | CC0 Creative Commons

Process Activities

Software Evolution

- Historically there has been a split between **software development** and **software evolution**
- This **distinction** is increasingly **irrelevant**
- Software is **usually not complete**
- Software engineering is rather **an evolutionary process**
- **Requirements change and so does software!**



Source: https://pixabay.com/en/evolution-walking-charles-darwin-297234/
by Clker-Free-Vector-Images | CC0 Creative Commons

Change

Coping with change

- Change is inevitable in software projects
- **Requirements change** (external pressures, management priorities, etc.)
- New technologies may lead to **new possibilities**

Source: https://pixabay.com/en/change-new-beginning-renewal-671374/
by geralt | CC0 Creative Commons

Two approaches to handle the cost of change:

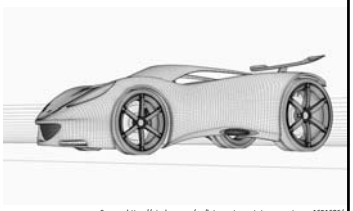
- **Change avoidance**
  The process activities are able to foresee change before rework is required
- **Change tolerance**
  Change can be integrated at relatively low cost (usually in incremental development)

MANAGEMENT CENTER INNSBRUCK

System prototyping

Prototypes are used to demonstrate concepts, test design options and find out about problems

- **Requirements engineering process**
  Elicitation and validation of requirements;
  New ideas;
  Find errors and omissions;
- **System design process**
  Explore possible solutions and UI designs;
  Check feasibility;
  Involve end-users;



Source: https://pixabay.com/en/lotus-auto-prototype-sports-car-1651606/ by PIRO4D |CC0 Creative Commons

MANAGEMENT CENTER INNSBRUCK

System prototyping

Objective of the prototype should be **made explicit** from the beginning (e.g. prototype the UI, validate functional requirements, demonstrate feasibility)

- **One prototype cannot serve all purposes!**
- Easy to get the wrong **type of feedback**
- Prototype is **not necessarily used the same way** as the end product
- **Training time** may be important
- Prototypes do **not always need to be functional** (e.g. paper prototypes)
- **"Wizard of Oz"** simulations can also be used

Source: https://pixabay.com/en/car-prototype-auto-automobile-1716088/ by MikesPhotos |CC0 Creative Commons

Change

MCI MANAGEMENT CENTER INNSBRUCK

Incremental Delivery

- Some of the developed **increments are delivered**
- Customers outline the **importance** of various services
- **High-priority** services are implemented **first**
- Increments are **completed and delivered**
- Customers are able to **use certain services earlier**
- New increments are **integrated with existing ones**

Source: https://pixabay.com/en/green-grass-prato-echo-ecological-1968590/ by ElisaRiva |CC0 Creative Commons

Change

MCI MANAGEMENT CENTER INNSBRUCK

Incremental Delivery - Advantages

- Customers can use early increments as **prototypes**
- Unlike prototypes, increments are **part of the real system**
- Customers **do not need to wait** until entire system is delivered
- It is easy to **incorporate change**
- Most important services **receive the most testing** (i.e. they are delivered first)

Source: https://pixabay.com/en/pro-yes-approve-accept-consent-1402153/ by johnhain |CC0 Creative Commons

MANAGEMENT CENTER INNSBRUCK

Incremental Delivery - Problems

Source: https://pixabay.com/en/thought-idea-innovation-imagination-2123970/ by TeroVesalainen |CCO Creative Commons

- It is hard to **identify common facilities** (code bases, functions, etc.) that are needed by all increments
- It is difficult if an **old system is being replaced**
  i.e. users expect the full functionality of the old system from the beginning
- It may conflict with **the procurement model** of many organizations
  i.e. they want full specification from the beginning

MANAGEMENT CENTER INNSBRUCK

Boehm's Spiral Model (Boehm. 1988)

Software process represented by a spiral where each loop represents a phase of he software process. Each loop splits in:

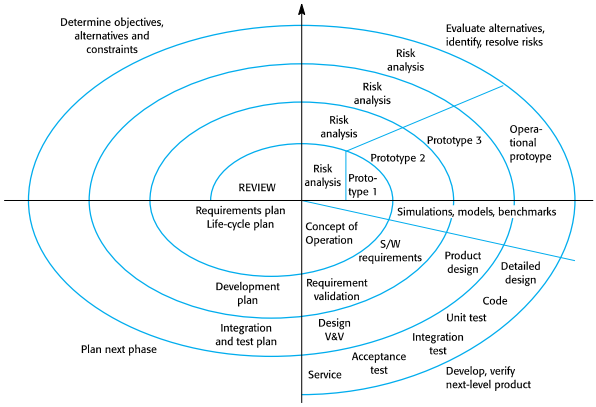
Source: https://pixabay.com/en/risk-word-letters-boggle-game-1945683/ by Wokandapix |CCO Creative Commons

- **Object setting**
  Objects and risks of the phase are defined;
- **Risk assessment and reduction**
  Risks analysis is carried out;
  Risk reduction steps are identified;
- **Development and validation**
  The specific development model of this phase is chosen based on the risk;
- **Planning**
  Project is reviewed and it is decided whether a new loop is needed

The main distinction of the spiral model is its explicit **recognition of risk**.

Change

Boehm's Spiral Model (Boehm. 1988)



Source: Sommerville, I. (2011). Software Engineering, 9th Ed. Boston, MA, USA: Pearson Education Inc.

Rational Unified Process

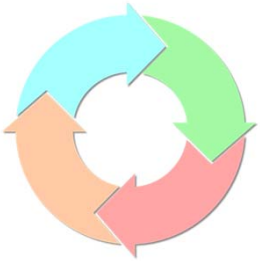
The Rational Unified Process (RUP)

(http://www-01.ibm.com/software/rational/rup/)

- Example of a modern process model derived from the work on **UML**
- Brings together **elements from all the generic process models**
- Illustrates **good practice in specification and design**
- Supports **prototyping and incremental delivery**

RUP is usually described from three perspectives

- A dynamic perspective
- A static perspective
- A practice perspective



Source: https://pixabay.com/en/cycle-phase-change-process-diagram-2019530/ by kmician | CC0 Creative Commons

MCI MANAGEMENT CENTER INNSBRUCK

The Rational Unified Process – dynamic perspective

More related to business rather than technical concerns.

RUP is divided into three phases:

- **Inception**
  Establish a **business case** for the system;
  Identify **people, systems and interactions**;
  Evaluate the potential **contribution** of the syste
- **Elaboration**
  Understand the **problem domain**;
  Establish an **architectural framework**;
  Develop a **project plan** and identify **risks**;
  Build a **requirements model**;
- **Construction**
  Design, program and test the system;

Source: https://pixabay.com/en/problem-analysis-solution-hand-67054/
by geralt |CCO Creative Commons

MCI MANAGEMENT CENTER INNSBRUCK

The Rational Unified Process – static perspective

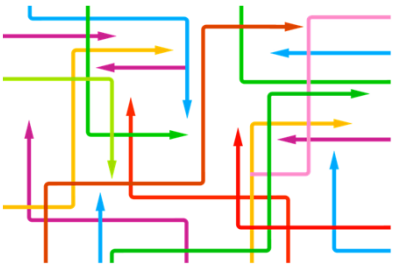
The static perspective of RUP focuses on workflows.

Core workflows

- Business modeling (business use cases)
- Requirements (actors and interactions)
- Analysis and design (system modeling)
- Implementation
- Testing
- Deployment

Support workflows

- Configuration and change management
- Project management
- Environment (software tools)



Source: https://pixabay.com/en/arrows-direction-production-planning-1577983/
by geralt |CCO Creative Commons

Rational Unified Process

The Rational Unified Process – practice perspective

The practice perspective describes good software engineering practices. Six fundamental practices are recommended.:

- **Develop software iteratively**
  i.e. plan increments based on priorities
- **Manage requirements**
  Explicitly document requirements
- **Use component-based architectures**
- **Visually model software (i.e. UML)**
- **Verify software quality**
- **Control changes to the software**

Source: https://pixabay.com/en/woman-girl-balloon-thought-bubble-1172718/ by geralt |CC0 Creative Commons

Summary

What you should have taken away from this class:

- Software processes are the **activities involved in producing a software system**. Software process **models are abstract representations of these processes**.
- General process **models describe the organization of software processes**. Examples include the **waterfall model**, **incremental development**, and **reuse-oriented development**.
- Requirements engineering is the process of developing a **software specification**. Design and implementation processes are concerned with **transforming a requirements specification into an executable software system**.
- **Software validation** is the **process of checking that the system conforms to its specification** and that it meets the real needs of the users of the system.
- **Software evolution** takes place when **you change existing software systems to meet new requirements.** Changes are continuous and the software must evolve to remain useful.
- Processes should **include activities to cope with change**. This may involve a **prototyping phase** that helps avoid poor decisions on requirements and design. Processes may be structured for **iterative development and delivery** so that changes may be made without disrupting the system as a whole.
- **The Rational Unified Process is a modern generic process model** that is organized into phases (i.e. inception, elaboration, and construction) but separates activities (requirements, analysis, and design, etc.) from these phases.

References

- Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. *IEEE Computer, 21(5)*, 61-72.
- Budgen, D. (2003). *Software Design (2nd Edition)*. Harlow, UK: Addison-Wesley.
- Massol, V., & Husted, T. (2003). *JUnit in Action.* Greenwich, CT, USA: Manning Publications Co.
- Rettig, M. (1994). Practical Programmer: Prototyping for Tiny Fingers. *Communications of the ACM, 37(4)*, 21-27.
- Royce, W. W. (1970). Managing the Development of Large Software Systems: Concepts and Techniques. *IEEE WESTCON*, (pp. 1-9). Los Angeles, CA, USA.
- Schmidt, D. C. (2006). Model-Driven Engineering. *IEEE Computer, 39(2)*, 25-31.

Thank you for your attention!

mentoring the motivated.