Chapter 2 – Software Processes

Lecture 1

Topics covered

- ✧ Software process models
- ✧ Process activities
- ✧ Coping with change
- ✧ The Rational Unified Process
  - An example of a modern software process.

The software process

- ✧ A structured set of activities used to develop a software system/product.
- ✧ Many different software processes but all involve:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking that it does what the customer wants;
  - Evolution – changing the system in response to changing customer needs.
- ✧ A software process model (or paradigm) is an abstract representation of a process
  - a framework that can be extended to create more specific processes, which are actually used to produce software

Software process descriptions

- ✧ Process descriptions may include process activities such as specifying a data model, designing a user interface, etc. and the ordering of these activities.
- ✧ Process descriptions may also include:
  - Products: the outcomes of a process activity (models, docs)
  - Roles: the responsibilities of the people involved in the process;
  - Pre- and post-conditions: statements that are true before and after a process activity has been enacted or a product produced.

Plan-driven and agile processes

✧ Processes often categorized as plan-driven or agile.

✧ Plan-driven processes:
- All of the process activities are planned in advance
- Progress is measured against this plan.

✧ Agile processes:
- Planning is incremental (occurs during different phases)
- It is easier to change the process to reflect changing customer requirements.

✧ In practice, most practical processes include elements of both plan-driven and agile approaches.

✧ Many organizations have their own software processes.

✧ There are no right or wrong software processes.

2.1 Software process models (frameworks)

✧ The waterfall model
- Plan-driven model. Separate and distinct phases of specification and development.

✧ Incremental development
- Specification, development and validation are interleaved, producing a series of versions. May be plan-driven or agile.

✧ Reuse-oriented software engineering
- The system is assembled from existing components. May be plan-driven or agile.

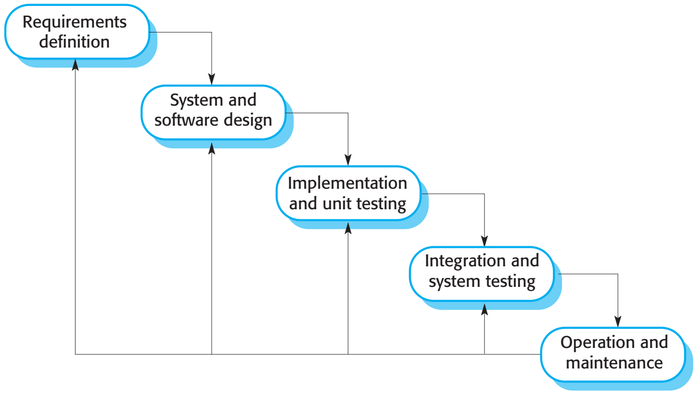
✧ In practice, most large systems are developed using a process that incorporates elements from all of these models.

Waterfall model phases

✧ There are separate identified phases in the waterfall model:
- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance

✧ Main drawback: The difficulty of accommodating change after the process is underway.
- In principle, a phase has to be complete before moving onto the next phase.
- Change requires "backtracking": revising previous step(s)

The waterfall model


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Waterfall model issues

✧ Partitioning the project into sequential stages makes it difficult to respond to changing customer requirements.
- Appropriate only when the requirements are well-understood and changes will be fairly limited during the design process.

✧ Can be used for large systems engineering projects where a system is developed at several sites.
- Plan-driven nature of the this model helps coordinate the work.

✧ Good for formal system development
- Mathematical model of system specifications is refined to programming language code using transformations
- Good when safety, reliability, and security requirements are critical.

Incremental development

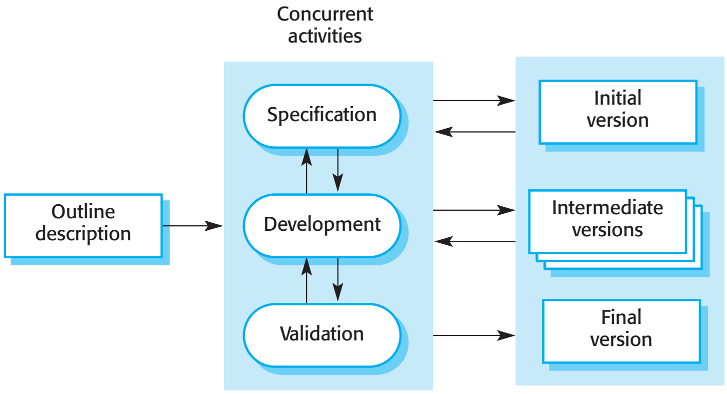
✧ Specification, development and validation are interleaved.

✧ The system is developed as a series of versions or releases (called increments).
- Each version adds functionality to the previous version

✧ Each version is exposed to the user for feedback

✧ Early versions can implement the most important, urgent, or risky features

Incremental development



Concurrent activities

Outline description

Specification

Development

Validation

Initial version

Intermediate versions

Final version

Incremental development benefits

✧ The cost of accommodating changing customer requirements is reduced.
- Analysis and documentation are added instead of reworked.

✧ It is easier to get customer feedback on the development work that has been done.
- Easier to present an incremental release than results of specification or design phase.

✧ Customers get functionality sooner.

✧ Can be plan-driven (versions are planned ahead) or agile (determine next increment as you go).

Incremental development problems

- The process is not visible.
  - generally less process documentation (for rapid development).
- System structure tends to degrade as new increments are added.
  - UNLESS time and money is spent on **refactoring** to improve the software.

  - Refactoring: disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.

Reuse-oriented software engineering

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Process stages
  - Requirements specification
  - Component analysis: search for close matches
  - Requirements modification: to reflect available components
  - System design with reuse: organize framework around acceptable components.
  - Development and integration: components are integrated along with new code
  - System validation

Types of software component

- Web services
  - Developed according to service standards
  - Are available for remote invocation.
- Collections of objects

  Developed as a package to be integrated with a component framework such as .NET or J2EE.
- Stand-alone software systems (COTS) that are configured for use in a particular environment.

Advantages and Disadvantages of Reuse-oriented Software Engineering

- Benefits
  - Reduces costs and risks (less code to write)
  - Usually leads to faster delivery.
- Disadvantages
  - Requirements may have to be compromised (no good matches)
  - Control over evolution of system is lost (dependent on developers of the components).

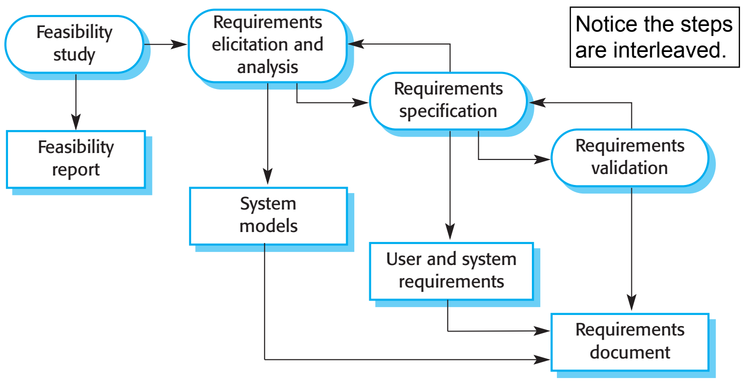
2.2 Process activities

- ✧ The four basic process activities:
  - specification
  - development
  - validation
  - evolution
- ✧ organized differently in different development processes. (i.e. in sequence or inter-leaved).
- ✧ Same activity may be carried out differently by different people, or different process methods (i.e. specifications can be typed into a document or written on cards).

Software specification

- ✧ The process of establishing:
  - what services are required (features) and
  - the constraints on the system's operation and development.
- ✧ Requirements engineering process
  - Feasibility study
    - Is it technically and financially feasible to build the system?
  - Requirements elicitation and analysis
    - What do the system stakeholders require or expect from the system?
    - May observe existing systems, develop models or prototype
  - Requirements specification
    - Defining the requirements in detail, write up in a document
  - Requirements validation
    - Checking the requirements for realism, consistency, and completeness.

The requirements engineering process

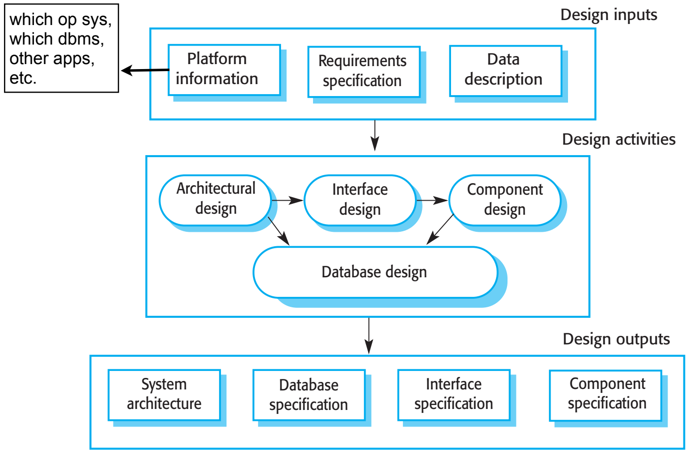


Notice the steps are interleaved.

Software design and implementation

- ✧ Converting the system specification into an executable system.
- ✧ Software design
  - Description of the structure of the software, data models, interfaces, algorithms, etc.
- ✧ Implementation
  - Translate the design into an executable program;
- ✧ Design and implementation are closely related and may be inter-leaved.

A general model of the design process



Design activities

- ✧ *Architectural design:* where you identify
  - the overall structure of the system,
  - the principal components,
  - their relationships and
  - how they are distributed.
- ✧ *Interface design,* where you precisely define the interfaces between system components (so they can be developed independently).
- ✧ *Component design,* where you design how each component will function (may be left up to developer).
- ✧ *Database design,* where you design the system data structures and how these are to be represented in a database.

Software validation

- ✧ Verification and validation (V & V) is intended to
  - show that a system conforms to its specification and
  - meets the requirements of the system customer.

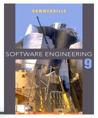
- ✧ Program testing is the principal validation technique. (executing the system over simulated data).

- ✧ Validation may also involve inspections and reviews

Testing stages

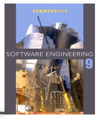
- ✧ Development or component testing
  - Individual components are tested independently by developers
  - Components may be functions or objects or coherent groupings of these entities.
  - Unit testing: JUnit is an automatic testing tool, can be re-run whenever the code is updated.
- ✧ System testing
  - Testing of the system as a whole (after integrating the components).
  - Especially looking for errors resulting from unanticipated interactions between components.
- ✧ Acceptance testing
  - Testing with customer data

Software evolution



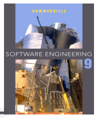
- ✧ Software is inherently flexible and can change (as opposed to hardware).
- ✧ Formerly, development and evolution were seen as two entirely separate processes:
  - development: creative, interesting.
  - evolution/maintenance: dull, easy
- ✧ Now development and maintenance are more fluid, interleaved: maintenance is just another increment, part of the original process.

Key points

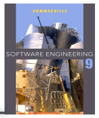


- ✧ Software processes are specific, structured sets of activities used to produce a software system.
- ✧ Software process models are abstract representations of these processes.
- ✧ General process models describe the organization or framework of software processes.
- ✧ Examples of these general models include
  - the ‘waterfall’ model,
  - incremental development, and
  - reuse-oriented development.

Key points



- ✧ 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. The software must evolve to remain useful.



Chapter 2 – Software Processes

Lecture 2

2.3 Coping with change

- Change is inevitable in all large software projects.
  - Business changes lead to new and changed system requirements
  - New technologies open up new possibilities for improving implementations
  - Changing platforms require application changes
- Change leads to rework:
  - new requirements lead to more requirements analysis
  - this may lead to redesign of the system or components
  - this may lead to changes to the implementation
  - this may lead to new tests, and re-testing the system

Reducing the costs of rework

- Change avoidance: include activities to anticipate possible changes before significant rework is required.
  - Develop a prototype to show some key features of the system to users, let them refine requirements before committing to them.
- Change tolerance: design process to accommodate change
  - Use incremental development.
  - Proposed changes may be implemented in new increments.
  - Or only a single old increment may have be changed.

Software prototyping

- A prototype is an initial version of a system used to demonstrate concepts and try out design options.
- Allows users to see how well systems supports their work, may lead to new ideas for requirements
- As prototype is developed, may reveal errors and omissions in the requirements
- Can check feasibility of design
  - For a database, make sure it efficient
  - For user interface, prototype is much better than a text description.

Prototype development process

- Objectives for prototype should be made in advance
- Decide what to put in, what to leave out.
- Let users test the prototype and evaluate it with respect to the objectives

Throw-away prototypes

- Prototypes should be discarded after development as they are not a good basis for a production system:
  - It may be impossible to tune the system to meet non-functional requirements;
  - Prototypes are normally undocumented;
  - The prototype structure is usually degraded through quick and dirty design;
  - The prototype probably will not meet normal organisational quality standards.

Incremental delivery

- The development AND delivery is broken down into increments: each increment is delivered to users.
- Each increment provides a subset of the required functionality as a separate release.
- Highest priority requirements are included in early increments.
- Requirements are frozen for the current increment, though requirements for later increments can continue to evolve.

Incremental delivery advantages

- Customer value can be delivered with each increment so system functionality is available earlier.
- Early increments act as a prototype to help elicit requirements for later increments.
- Like incremental development, it should be relatively easy to incorporate change.
- The highest priority system services tend to receive the most testing.

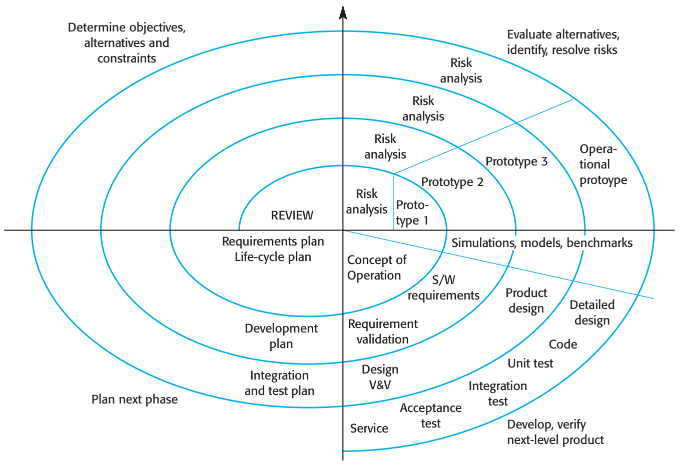
Incremental delivery problems

- It can be difficult to identify/specify common facilities that are needed by all increments.
- The specification is not complete until final increment.
  - This conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.
- Difficult to replace an existing system as increments have less functionality than the system being replaced.

Boehm's spiral model

- ✧ Risk driven process framework
- ✧ Process is represented as a spiral.
- ✧ Each loop in the spiral represents a phase in the process.
- ✧ No fixed phases such as specification or design - loops in the spiral are chosen depending on elements of risk.
- ✧ Risks are explicitly assessed and resolved throughout the process.

Boehm's spiral model of the software process



Spiral model sectors

- ✧ Objective setting
  - Specific objectives for the phase are identified.
- ✧ Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce the key risks.
- ✧ Development and validation
  - A development model for the system is chosen which can be any of the generic models, appropriate for current risk
- ✧ Planning
  - The project is reviewed and the next phase of the spiral is planned.

Spiral model usage

- ✧ Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.
- ✧ In practice, however, the model is rarely used as published for practical software development.

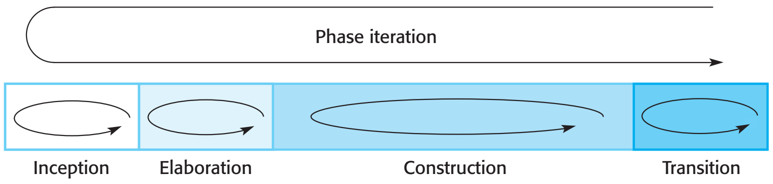
2.4 The Rational Unified Process

- ✧ A modern generic process derived from the work on the UML and associated process.
- ✧ Brings together aspects of the 3 generic process models discussed previously.
- ✧ Normally described from 3 perspectives
  - A dynamic perspective that shows phases over time;
  - A static perspective that shows process activities;
  - A practice perspective that suggests good practice.

RUP phases

- ✧ Inception
  - Establish the business case for the system. Who uses it? what do they get out of it?
- ✧ Elaboration
  - Develop an understanding of the problem domain and develop the system architecture, develop plan, identify risk.
- ✧ Construction
  - System design, programming and testing.
- ✧ Transition
  - Deploy the system in its operating environment.

Phases in the Rational Unified Process



Note the phases may be repeated iteratively/incrementally

Static workflows (process activities) in the Rational Unified Process

Note: workflows are not tied to specific phases

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.

Static workflows (process activities) in the Rational Unified Process

Workflow	Description
Testing	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system (see Chapter 25).
Project management	This supporting workflow manages the system development (see Chapters 22 and 23).
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

RUP good practice

- ✧ Develop software iteratively
  - Plan increments based on customer priorities and deliver highest priority increments first.
- ✧ Manage requirements
  - Explicitly document customer requirements and keep track of changes to these requirements.
- ✧ Use component-based architectures
  - Organize the system architecture as a set of reusable components.

RUP good practice

- ✧ Visually model software
  - Use graphical UML models to present static and dynamic views of the software.
- ✧ Verify software quality
  - Ensure that the software meet’s organizational quality standards.
- ✧ Control changes to software
  - Manage software changes using a change management system and configuration management tools.

Key points

- ✧ 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 (inception, elaboration, construction and transition) but separates activities (requirements, analysis and design, etc.) from these phases.