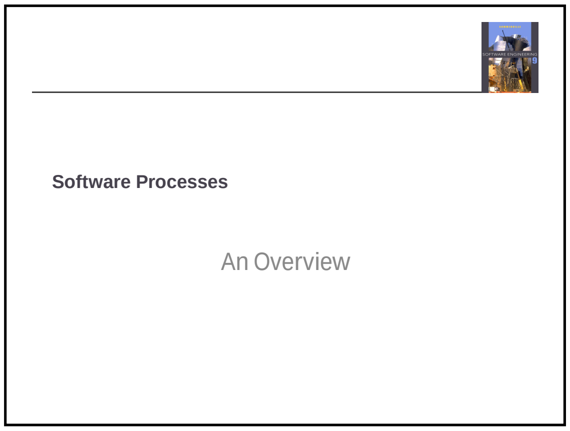
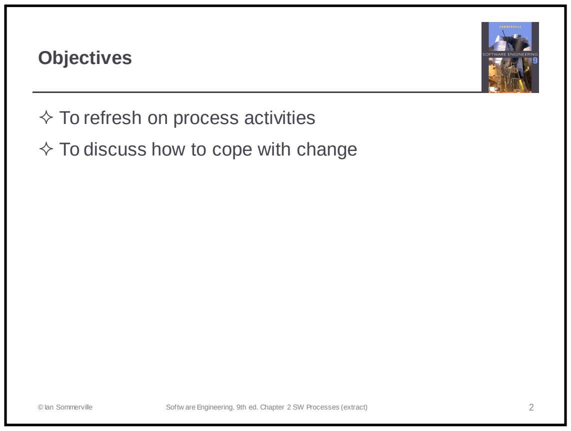


Software Processes

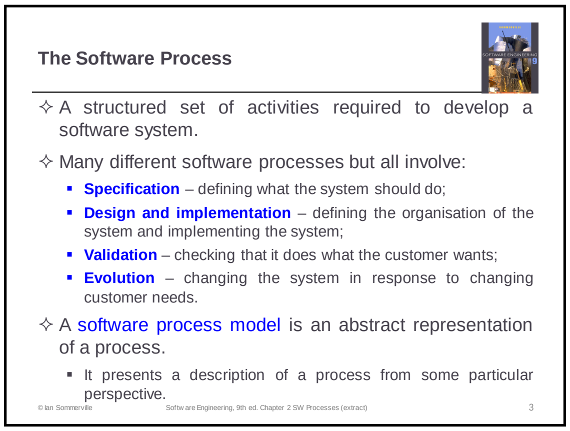
An Overview



Objectives

✧ To refresh on process activities

✧ To discuss how to cope with change



The Software Process

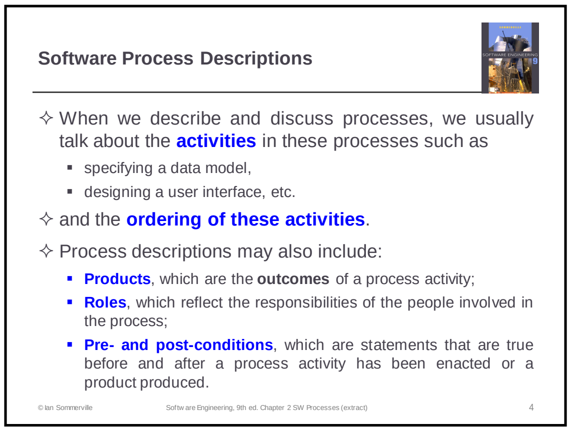
✧ A structured set of activities required to develop a software system.

✧ Many different software processes but all involve:

- **Specification** – defining what the system should do;
- **Design and implementation** – defining the organisation 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 is an abstract representation of a process.

- It presents a description of a process from some particular perspective.



Software Process Descriptions

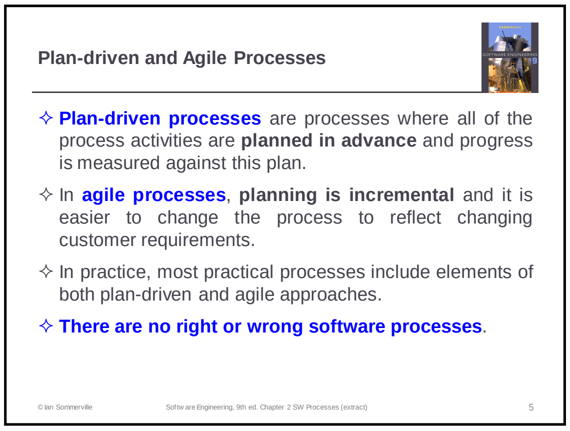
✧ When we describe and discuss processes, we usually talk about the **activities** in these processes such as

- specifying a data model,
- designing a user interface, etc.

✧ and the **ordering of these activities**.

✧ Process descriptions may also include:

- **Products**, which are the **outcomes** of a process activity;
- **Roles**, which reflect the responsibilities of the people involved in the process;
- **Pre- and post-conditions**, which are statements that are true before and after a process activity has been enacted or a product produced.



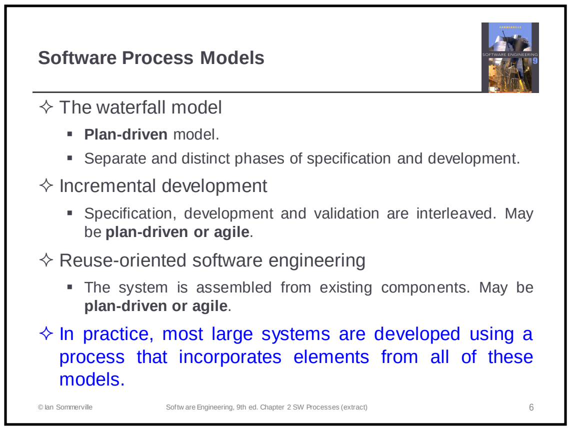
Plan-driven and Agile Processes

✧ **Plan-driven processes** are processes where all of the process activities are **planned in advance** and progress is measured against this plan.

✧ In **agile processes**, **planning is incremental** and 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.

✧ **There are no right or wrong software processes**.



Software Process Models

✧ The waterfall model

- **Plan-driven** model.
- Separate and distinct phases of specification and development.

✧ Incremental development

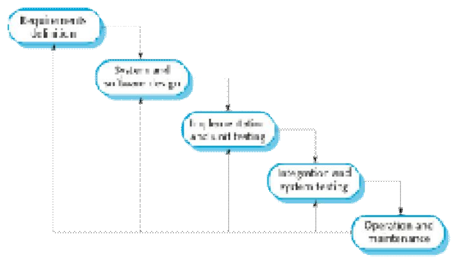
- Specification, development and validation are interleaved. May be **plan-driven or agile**.

✧ Reuse-oriented software engineering

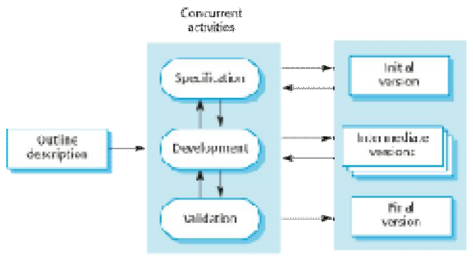
- The system is assembled from existing components. May be **plan-driven or agile**.

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

The Waterfall Model


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

Incremental Development



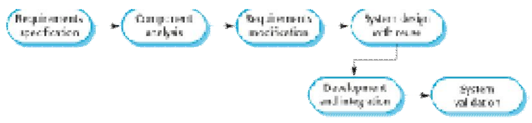
Reuse-oriented Software Engineering

✧ Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.

✧ Process stages
- Component analysis;
- Requirements modification;
- System design with reuse;
- Development and integration.

✧ **Reuse is now the standard approach for building many types of business system**

Reuse-oriented Software Engineering


Requirements specification
Component analysis
Requirements modification
System design with reuse
Development and integration
System validation

Types Of Software Component

✧ Web services that are developed according to service standards and which are available for remote invocation.

✧ Collections of objects that are 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.

Process Activities

✧ Real software processes are inter-leaved sequences of **technical, collaborative and managerial activities** with the overall goal of specifying, designing, implementing and testing a software system.

✧ The four basic process activities are organised differently in different development processes.
- specification, development, validation and evolution

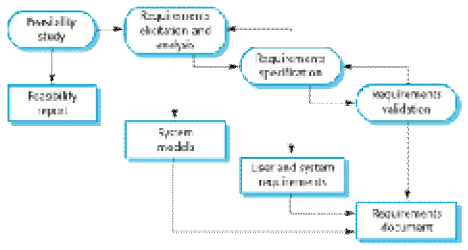
✧ In the waterfall model, they are organised in sequence, whereas in incremental development they are inter-leaved.

Software Specification

✧ The process of establishing what services are required 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?
- Requirements specification
  - Defining the requirements in detail
- Requirements validation
  - Checking the validity of the requirements

The Requirements Engineering Process



Software Design & Implementation

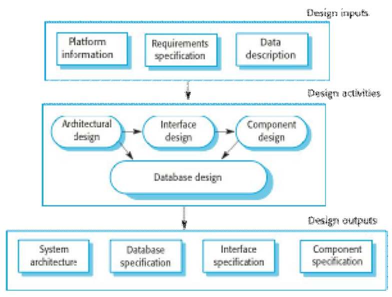
✧The process of **converting the system specification into an executable system**.

✧Software **design**
- Design a software structure that realises the specification;

✧**Implementation**
- Translate this structure into an executable program;

✧The activities of 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 (sometimes called sub-systems or modules), their relationships and how they are distributed.

✧ ***Interface design***, where you define the interfaces between system components.

✧ ***Component design***, where you take each system component and design how it will operate.

✧ ***Database design***, where you design the system data structures and how these are to be represented in a database.

Software Validation

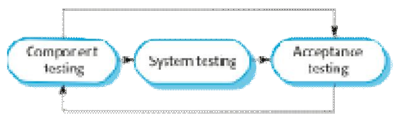
✧ **Verification and validation** (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.

✧ Involves checking and review processes and system testing.

✧ **System testing** involves executing the system with **test cases** that are derived from the specification of the real data to be processed by the system.

✧ Testing is the most commonly used V & V activity.

Stages of testing



Testing stages

✧ **Development or component testing**
- Individual components are tested independently;
- Components may be functions or objects or coherent groupings of these entities.

✧ **System testing**
- Testing of the system as a whole.
- Testing of **emergent properties** is particularly important.
  - i.e. the *properties of the system as a whole* rather than properties that can be derived from the properties of components of a system
  - are a consequence of the relationships between system components
  - can therefore only be assessed and measured once the components have been integrated into a system

✧ **Acceptance testing**
- Testing with customer data to check that the system meets the customer's needs.

Examples of Emergent Properties

✧ **The overall weight of the system**
- This is an example of an emergent property that can be computed from individual component properties.

✧ **The reliability of the system**
- This depends on the reliability of system components and the relationships between the components.

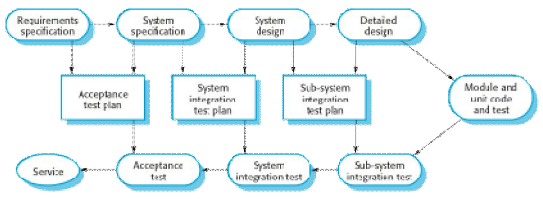
✧ **The usability of a system**
- This is a complex property which is not simply dependent on
  - the system **hardware** and **software** but also depends on
  - the system **operators** and
  - the **environment** where it is used.

Types of Emergent Properties

✧ **Functional properties**
- These appear when all the parts of a system work together to achieve some objective.
  - a bicycle has the functional property of being a transportation device once it has been assembled from its components.

✧ **Non-functional emergent properties**
- Examples are reliability, performance, safety, and security.
- These relate to the behaviour of the system in its operational environment.
- They are often critical for computer-based systems as failure to achieve some minimal defined level in these properties may make the system unusable.

Testing Phases In A Plan-driven Software Process


Requirements specification
System specification
System design
Detailed design
Acceptance test plan
System integration test plan
Sub-system integration test plan
Module and unit code and test
Service
Acceptance test
System integration test
Sub-system integration test

Software evolution

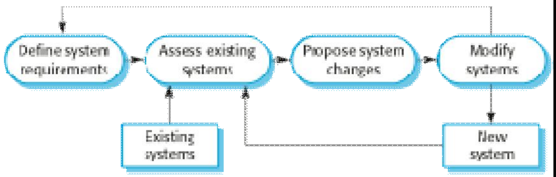
✧ Software is inherently flexible and can change.

✧ As requirements change through changing business circumstances, the software that supports the business must also evolve and change.

✧ In the past development and evolution (maintenance) were demarcated.

✧ Today this demarcation is increasingly irrelevant as fewer and fewer systems are completely new.

System evolution



Key points

✧ **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 organisation 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.

Coping With Change

✧ **Change is inevitable** in all large software projects.

✧ Q: What necessitates software change?
- 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** so the costs of change include
- both rework (e.g. re-analysing requirements) as well as
- the costs of implementing new functionality

Reducing The Costs Of Rework

✧ **Change avoidance**
- The software process includes activities that can **anticipate possible changes** before significant rework is required.
  - For example, a **prototype system** may be developed to show some key features of the system to customers.

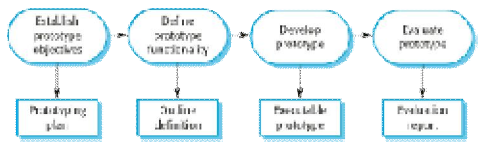
✧ **Change tolerance**
- The process is designed so that **changes can be accommodated** at relatively low cost.
- This normally involves some form of **incremental development**.
- Proposed changes may be implemented in increments that have not yet been developed.
- If this is impossible, then only a single increment (a small part of the system) may have be altered to incorporate the change.

Software Prototyping…a reminder

✧ A prototype is an initial version of a system used to demonstrate concepts and try out design options.

✧ A prototype can be used in:
- The **requirements engineering process** to help with requirements elicitation and validation;
- In **design processes** to explore options and develop a UI design;
- In the **testing process** to run back-to-back tests.

The process of prototype development



Incremental Delivery...a reminder

✧ Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with **each increment delivering part of the required functionality**.

✧ User requirements are prioritised and the highest priority requirements are included in early increments.

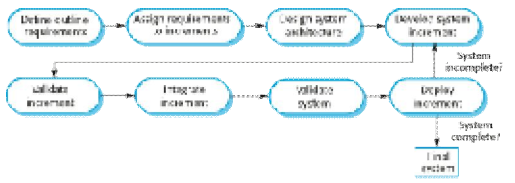
✧ Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

Incremental development and delivery

✧ Incremental development
- Develop the system in increments and evaluate each increment before proceeding to the development of the next increment;
- Normal approach used in **agile methods**;
- Evaluation done by user/customer proxy.

✧ Incremental delivery
- Deploy an increment for use by end-users;
- More realistic evaluation about practical use of software;
- Difficult to implement for replacement systems as increments have less functionality than the system being replaced.

Incremental delivery



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.