Software Process Models.

You will learn about a number of alternate generic models for structuring the software development process. The advantages and disadvantages of each. And when it is appropriate to use each model.

- The waterfall model.
- Prototyping.
- Incremental model.
- Spiral model.
- Iterative model.
- Agile model.
- Reuse-oriented software Engineering

The Software Process

- A software process is a set of related activities that leads to the production of the software. Any software process must include the following four activities:
  - **Software specification** (or requirements engineering): Define the main functionalities of the software and the constrains around them.
  - **Software design and implementation**: The software is to be designed and programmed.
  - **Software verification and validation**: The software must conforms to it's specification and meets the customer needs.
  - **Software evolution** (software maintenance): The software is being modified to meet customer and market requirements changes.

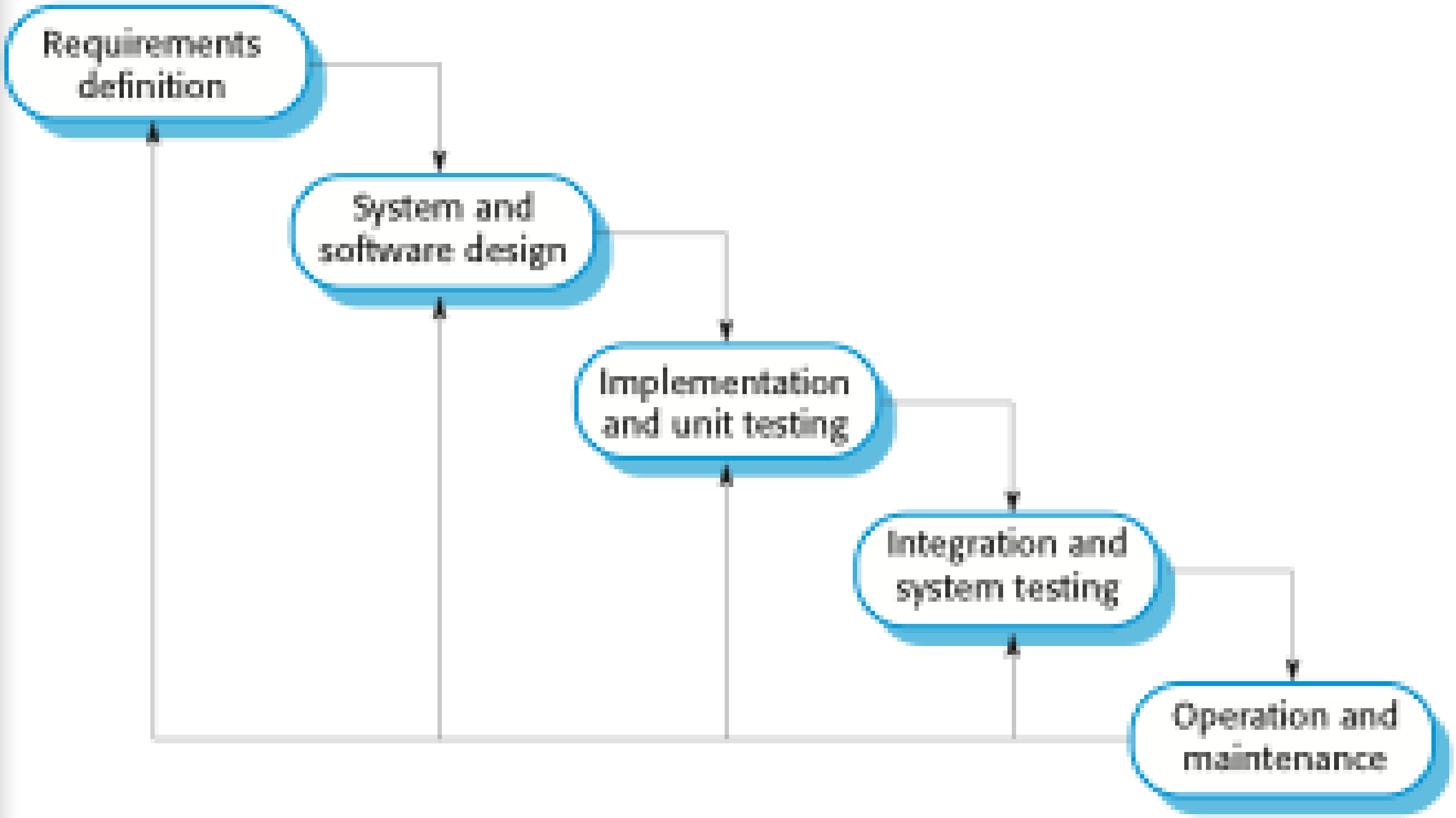
Software Process Models

- A software process model is a simplified representation of a software process. Each model represents a process from a specific perspective.
  - prescribes the order and frequency of phases.
  - specifies criteria for moving from one phase to the next.
  - defines the deliverables of the project.

The Waterfall Process Model (1970)

- One of the oldest software process models. Despite its many weaknesses it is still widely used and is the basis for many other more effective processes.
  - The waterfall model is a sequential approach, where each fundamental activity of a process is represented as a distinct separate phase, arranged in linear order.
  - The output of one phase is the input for the next, a phase is not started until the previous is complete.
  - Milestones are reached if the appropriate documentation is delivered (e.g. requirements specification, design specification, program, test document), document heavy. Often contracts are signed.
  - In the waterfall model, you must plan and schedule all of the activities before starting work (plan-driven process). *Plan-driven process is a process where all the activities are planned first, and the progress is measured against the plan. While with an agile process, planning is incremental and it's easier to change the process to reflect requirement changes.*

Waterfall Model


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

Each phase has to be completed before going on to the next.

The Nature of Waterfall Phases

- In principle, the result of each phase is one or more documents that should be approved and the next phase shouldn't be started until the previous phase has completely been finished.
- In practice, however, these phases overlap and feed information to each other. For example, during design, problems with requirements can be identified, and during coding, some of the design problems can be found, etc.
- The software process therefore is not a simple linear but involves feedback from one phase to another. So, documents produced in each phase may then have to be modified to reflect the changes made.

Waterfall Model Advantages

- Simple and easy to use. Phases are carried out and completed sequentially with specific inputs and deliverables.
- Practised for many years and there is a lot of experience in its use.
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Reinforces good habits define-before-design, design-before-code
- Facilitates allocation of resources – staff with phase specific skills.
- Works well for projects where requirements are well understood.

Waterfall Model Disadvantages

- Real projects rarely follow a sequential approach the software process is not a simple linear model but requires iteration.
- Requirements must be known up front it is unrealistic to expect accurate requirements at the start of the project.
- Hard to estimate reliably - estimates become more precise as the project progresses
- Testing occurs at the end of development, the first time the system is tested as a whole.
- Software is delivered late in the project- there are no intermediate versions so discovery of serious errors is delayed.

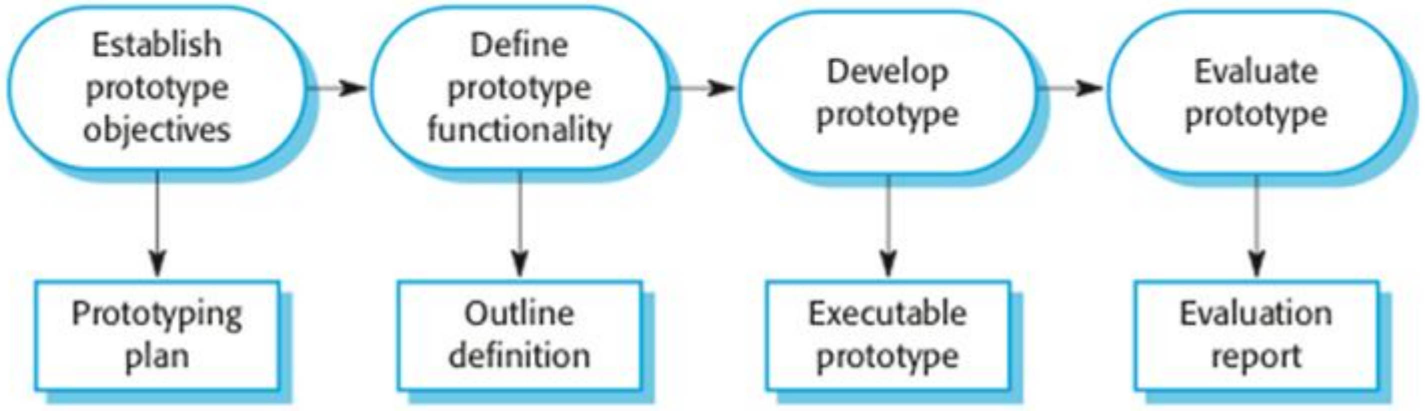
Applicability

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements. Consequently the waterfall model should only be applied when requirements are **well understood** and **unlikely to change radically** during development as this model has a relatively rigid structure which makes it hard to accommodate change when the process in underway.
- The waterfall model is mostly used for systems engineering projects where a system may be developed at several sites. In those circumstances, the plan-driven nature of the waterfall model helps coordinate the critical systems which need detailed, precise and accurate documentatiom.

Prototyping

- A prototype is a version of a system or part of the system that's developed quickly to check the customer's requirements or feasibility of some design decisions.
- A prototype is useful when a customer or developer is not sure of the requirements, algorithms, efficiency, business rules, response time, etc.
- Requires user involvement, which increases the likelihood of acceptance of the final implementation.
- While some prototypes are developed with the expectation that they will be discarded (throw away), it is possible in some cases to evolve from prototype to working system (evolutionary).

Phases of A Prototype



Throw-away Prototyping Approach

- **Establish objectives**: The objectives of the prototype should be made explicit from the start of the process. Is it to validate system requirements, or demonstrate feasibility, etc.
- **Define prototype functionality**: Decide what are the inputs and the expected output from a prototype. To reduce the prototyping costs and accelerate the delivery schedule, some functionality, such as response time and memory utilisation may be ignored unless they are relevant to the objective of the prototype.
- **Develop the prototype**: The initial prototype is developed that includes only user interfaces.
- **Evaluate the prototype**: Once the users are trained to use the prototype, they then discover requirements errors. Using the feedback both the specifications and the prototype can be improved. If changes are introduced, then a repeat of steps 3 and 4 may be needed.

Strengths and Problems with Throwaway Prototyping

- Strengths
  - Balance the benefits of well thought out analysis and design phases with the advantage of using prototypes to examine key issues. The resulting system will be more stable than that developed using system prototyping.
- Problems
  - It can take longer to deliver the final system will consequently add to the expense of the project.
  - Good tools need to be used for quick development in order to complete a prototype. In addition, the costs for training the development team on prototyping may be high.

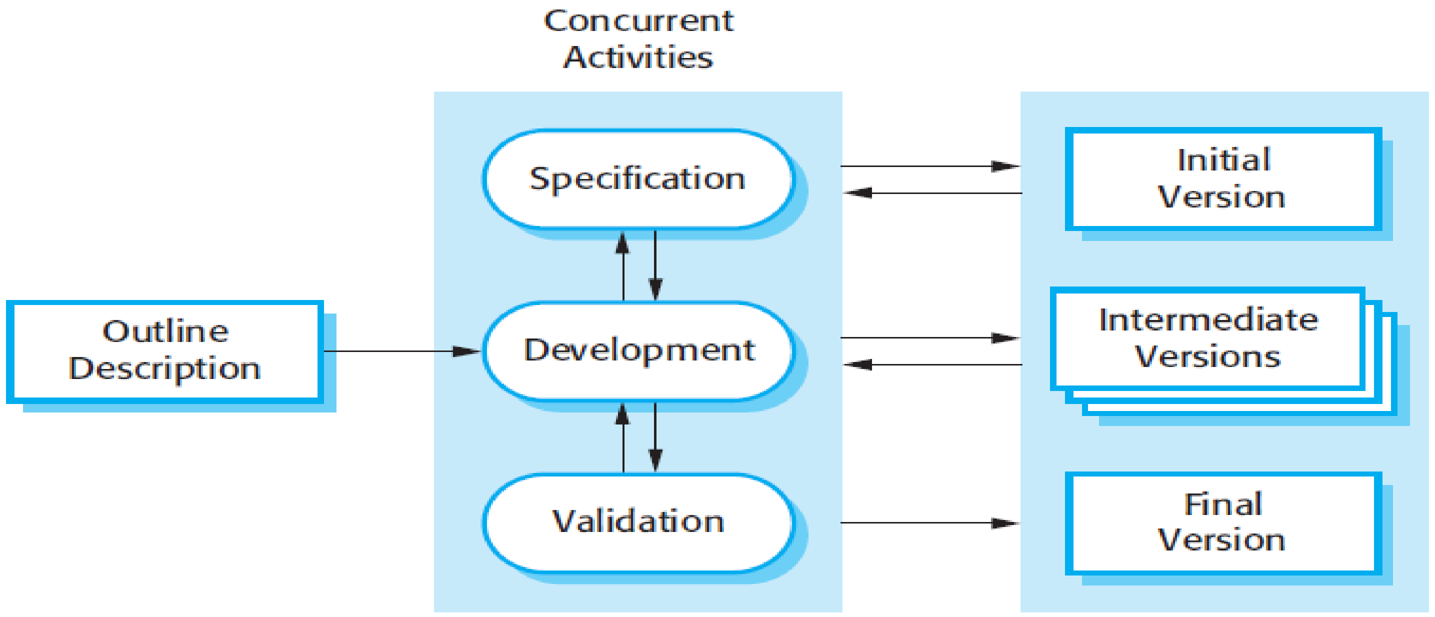
Applicability

- Prototyping is generally not a standalone, complete development process, but rather an approach to be used in the context of a full process (such as waterfall, incremental, spiral, etc).

Incremental Development

- Incremental development is based on the idea of developing an initial implementation, exposing this to user feedback, and evolving it through several versions until an acceptable system has been developed.
- Customers identify the services to be provided by the system and prioritise these. A number of delivery increments are then defined each providing a subset of the system functionality.
- Allocation of services depends on the service priority the highest priority are delivered first.
- Once increments have been identified the requirements for the services to be delivered in the first increment are defined in detail and that increment is developed.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.
- Once an increment is completed and delivered, customers can start to use it thus taking early delivery of system functionality.

Incremental Development



The activities of a process are not separated but interleaved with feedback involved across those activities.

Incremental Development Advantages

- Customer value can be delivered with each increment.
- Accelerated delivery of customer services.
- Highest priority services are delivered first and later increments are integrated with them, thus undergoing the most testing – less failures in the most important part of the system.
- Early increments act as a prototype to help elicit requirements for later increments.
- Lower risk of project failure

Incremental Development Disadvantages

- Some organisations have procedures that have evolved over the time, and can't follow informal incremental process. For example, procedures to ensure that the software properly implements external regulations.
- Increments should be small and yet deliver some functionality. It may be difficult to map functionality onto increments of the right size.
- Specification develops with the software. This conflicts with the complete specification forming part of the development contract. In the incremental approach there is no complete specification until the last increment is delivered.

Incremental Development

- Incremental development is one of the most common approaches. This approach can be either plan-driven or agile, or both.
- In a plan-driven approach, the system increments are identified in advance, but, in the agile approach, only the early increments are identified and the development of later increments depends on the progress and customer priorities.

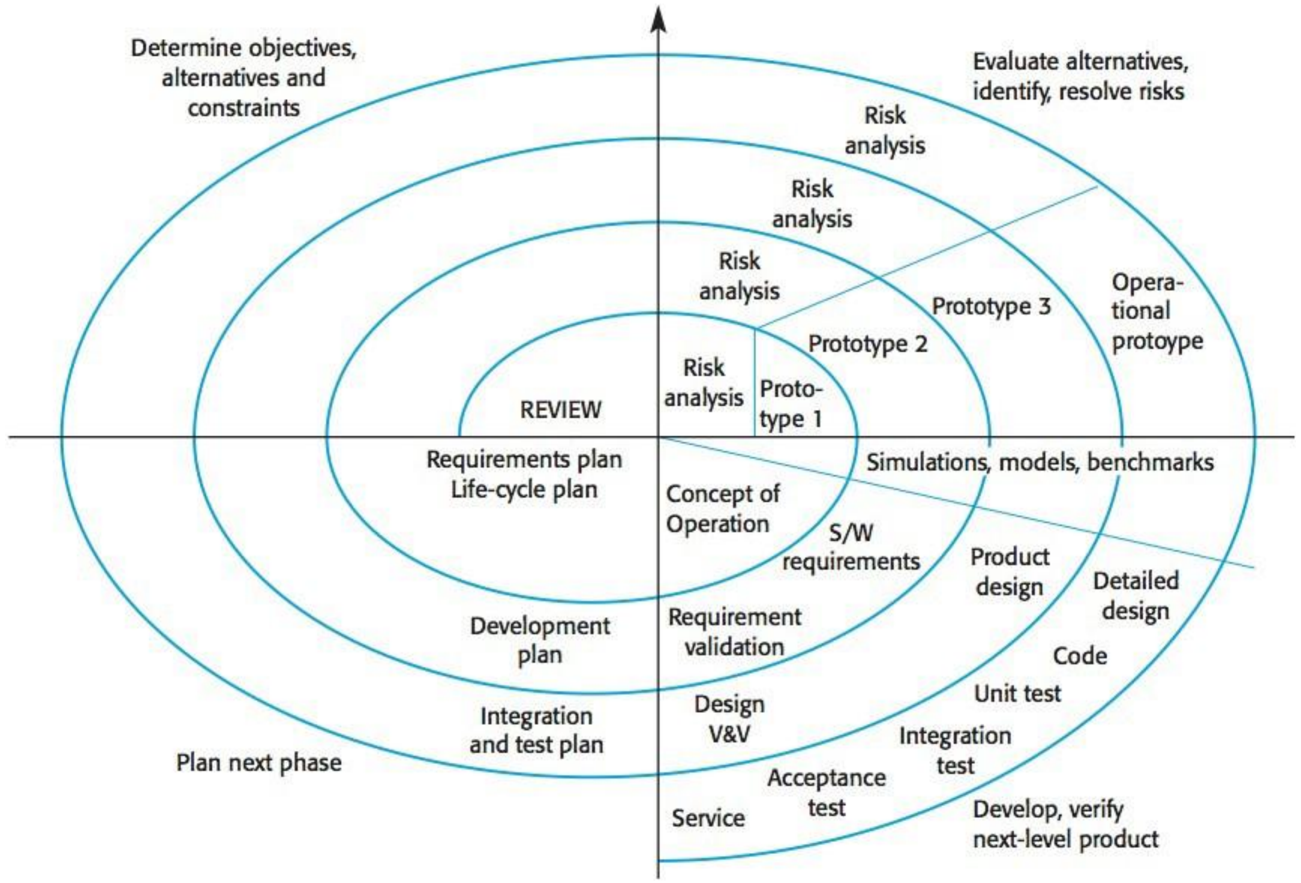
Incremental Vs Waterfall

- Incremental software development is better than a waterfall approach for most business, e-commerce, and personal systems.
- Compared to the waterfall model, incremental development has three important benefits:
  - The **cost of accommodating changing** customer requirements is reduced. The amount of analysis and documentation that has to be redone is much less than that required with waterfall model.
  - It is easier to get **customer feedback** on the work done during development than when the system is fully developed, tested, and delivered.
  - More **rapid delivery** of *useful* software is possible even if all the functionality hasn't been included. Customers are able to use and gain value from the software earlier than it's possible with the waterfall model.

Spiral Development

- The spiral model is similar to the incremental model, with more emphasis placed on risk analysis.
- The spiral model is risk-driven where the process is represented as spiral rather than a sequence of activities.
- It was designed to include the best features from the waterfall and prototyping models, and introduces a new component; risk-assessment.
- Each loop in the spiral represents a phase. Thus the first loop might be concerned with system feasibility, the next loop might be concerned with the requirements definition, the next loop with system design, and so on.
- Risks are explicitly assessed and resolved throughout the process. Risk-driven process

Spiral Model Boehm B. W.



Spiral Model

- Each loop in the spiral is split into four sectors:
  - **Objective setting:** The objectives and risks for that phase of the project are defined.
  - **Risk assessment and reduction:** For each of the identified project risks, a detailed analysis is conducted, and steps are taken to reduce the risk. For example, if there's a risk that the requirements are inappropriate, a prototype may be developed.
  - **Development and validation:** After risk evaluation, a process model for the system is chosen. So if the risk is expected in the user interface then we must prototype the user interface. If the risk is in the development process itself then use the waterfall model.
  - **Planning:** The project is reviewed and a decision is made whether to continue with a further loop or not.
- 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.

Spiral Model Advantages

- Risks are managed early and throughout the process – they are reduced before they become problematic. High amount of risk analysis.
- Software evolves as the project progresses – it is a realistic approach to the development of large-scale software. Errors and unattractive alternatives are eliminated early. Good for large and mission-critical projects.
- Planning is built into the process – each cycle contains a planning step to help monitor and keep a project on track
- Software is produced early in the software life cycle.

Spiral Model Disadvantages

- Can be a costly model to use
- Complicated to use requires risk assessment expertise.
- Project's success is highly dependent on risk analysis phase
- Doesn't work well for smaller projects. May be overkill for small projects – does not make sense if the cost of risk analysis is a major part of the overall cost.

Iterative Development

- Iterative development model aims to develop a system through building small portions of **all** the features, across **all** components.
- A product which meets the initial scope and released quickly for customer feedback. An early version with limited features important to establish market and get customer feedback.

Iterative Development Advantages

- A high-level design of the application is created before actually building the product and defining the design solution for the entire product.
- The product is being built and improved step by step hence defects can be tracked at early stages.
- Reliable user feedback can be obtained. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.
- Less time is spent on documenting and more time is given for designing.

Iterative Development Disadvantages

- Each phase of iteration is rigid with no overlaps.
- Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle.

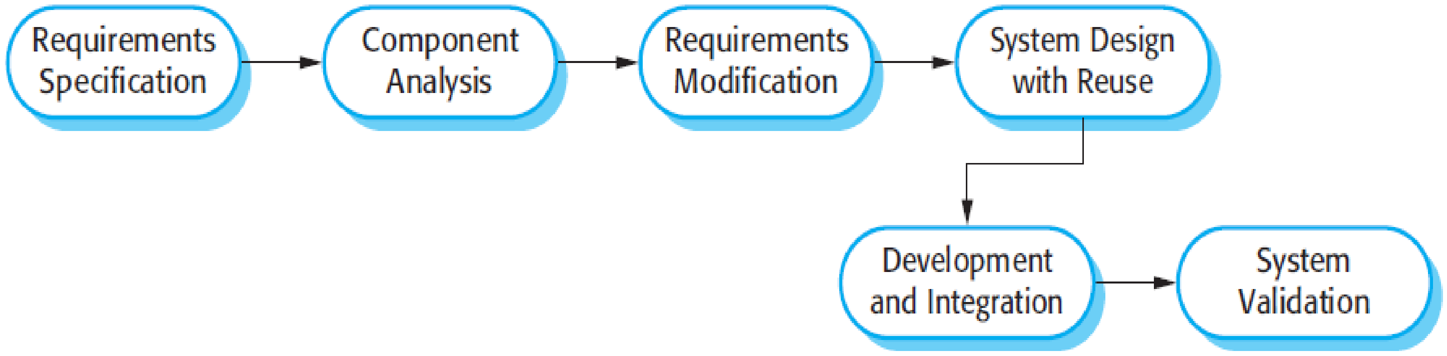
Agile

- A group of software development models based on the incremental and iterative approach suits small-medium size project with changing requirements.
- Customers are involved in the development.
- Documentation is minimised by using informal communications rather than formal meetings with written documents.
- Limited planning is required to get started with the project. Daily meetings make it possible to measure productivity.
- Team must be highly skilled.

Increment Vs Iterative Vs Agile

- Each increment in the incremental approach builds a complete feature of the software, while in iterative, it builds small portions of all the features.
- An agile approach combines the incremental and iterative approach by building a small portion of each feature, one by one, and then both gradually adding features and increasing their completeness.

Reuse-oriented Software Engineering


Requirements Specification
Component Analysis
Requirements Modification
System Design with Reuse
Development and Integration
System Validation

Attempts to reuse an existing design or code (probably also tested) that is similar to what is required. It is then modified, and incorporated to the new system.

Reuse-oriented Software Engineering

- The initial "requirements specification" phase and the "validation " phase are comparable with other software processes, intermediate phases in a reuse-oriented process are different.
- **Component analysis:** A search is made for the components to implement the given requirements specification. Usually, there's no exact match, and components may only provide some of the functionality required.
- **Requirements modification**: During this phase, the requirements are analysed using information about the components that have been discovered. They are then modified to reflect the available components. If the modifications are impossible, the *component analysis* activity may be re-entered to search for alternative solutions.

Reuse-oriented Software Engineering

- **System design with reuse:** During this phase, the framework of the system is designed or an existing framework is reused. The designers take into account the components that are reused and they will organise the framework accordingly. Some new software has to be designed if some reusable components are not available.
- **Development and integration:** The components are integrated to create the new system. System integration, in this model, may be part of the development process rather than a separate activity.

Reuse-oriented Software Engineering

Advantages

- Reduces the amount of software to be developed and so reduces costs and risks.
- Leads to faster delivery and deployment of the system.

Disadvantages

- Leads to requirements compromise and system may not meet user requirements.
- Control over system evolution is lost as new versions of the components are not under the control of the organisation using them.

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.
  - Management priorities change.
  - Changes from regulator.
- 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, where 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.

Reducing the costs of rework

- Change tolerance, where 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.

Summary

- A software process describes the interrelationships among the phases by expressing their order and frequency, as well as defining deliverables.
- Specific software processes are called software process models.
- One size does not fit all.