Software Lifecycle and Processes

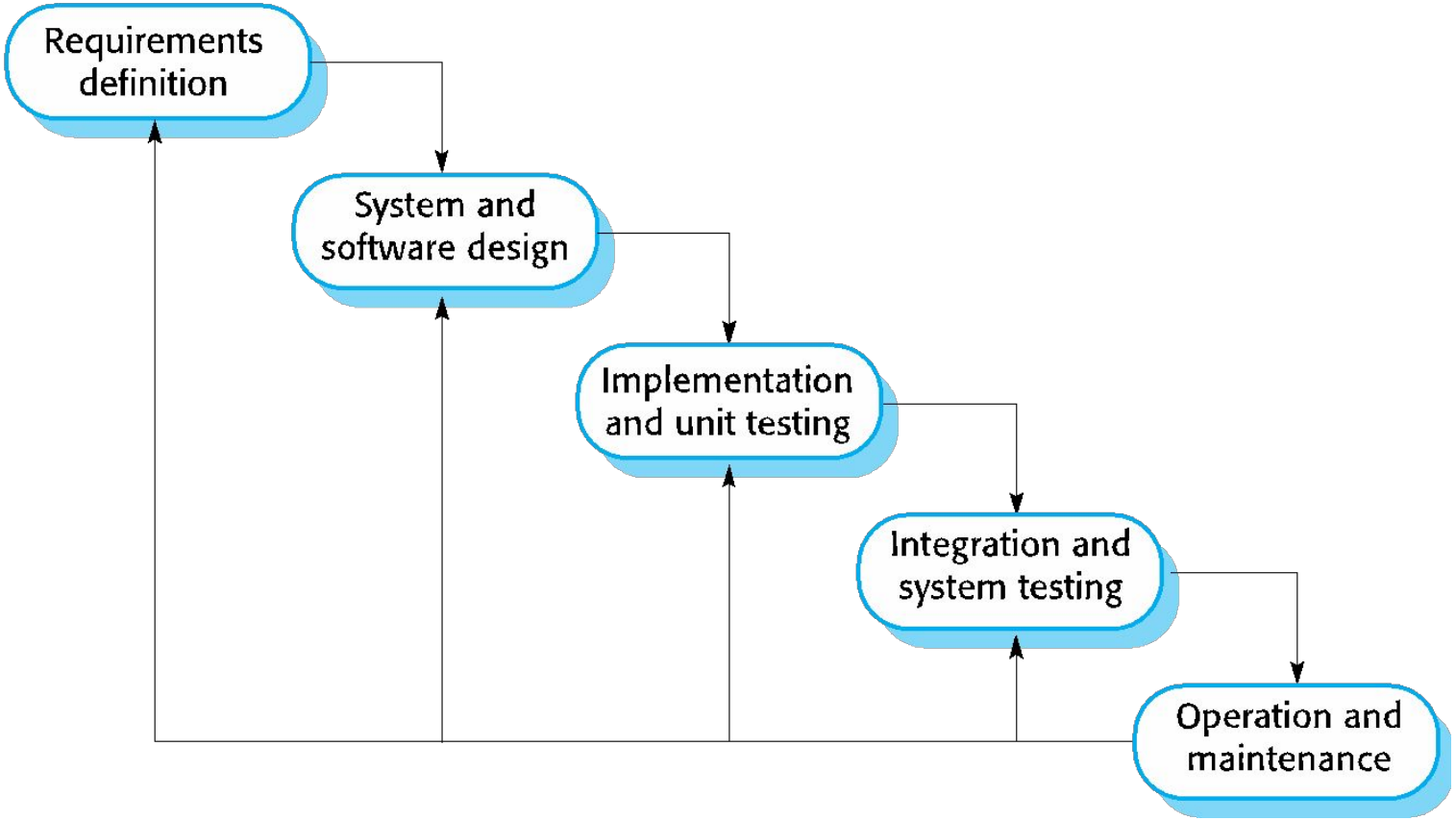
The software process

- 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 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.

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.
- Integration and configuration
  - The system is assembled from existing configurable components. May be plan-driven or agile.
- …

The waterfall model



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
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway.
  - A phase has to be complete before moving onto the next

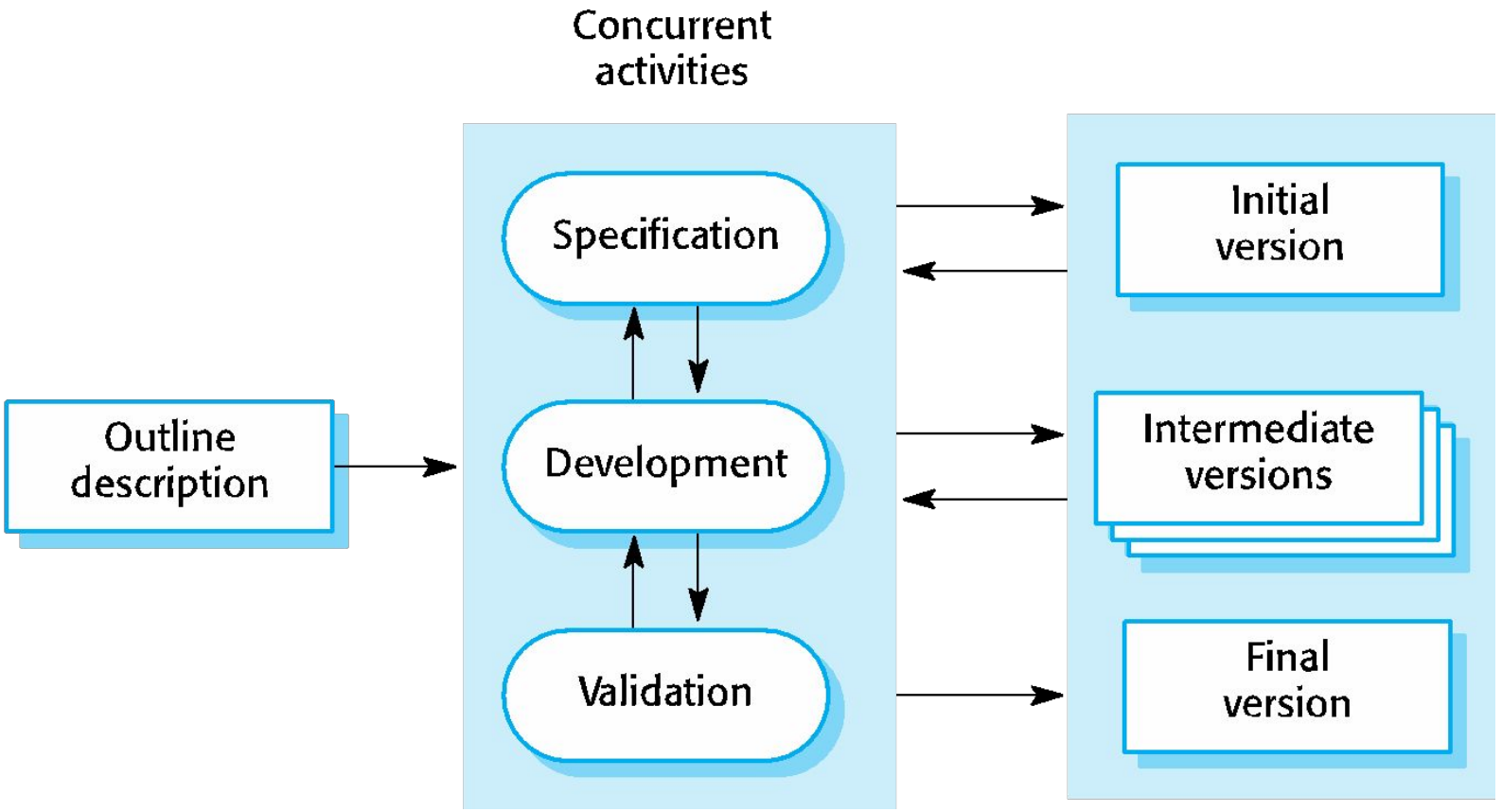
Waterfall model problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Incremental development

- 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



Incremental development could be Plan-driven and agile approaches

- system increments are identified

- early increments are identified, later increments depend on progress and customer priorities.

Incremental development 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 is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
  - Customers can comment on demonstrations of the software and see how much has been implemented.
- More rapid delivery and deployment of useful software to the customer is possible.
  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Integration and configuration

- Based on software reuse where systems are integrated from existing components or application systems
- Reused elements may be configured to adapt their behaviour and functionality to a user's requirements
- Reuse is now the standard approach for building many types of business system
  - Reuse covered in more depth in Chapter 15.

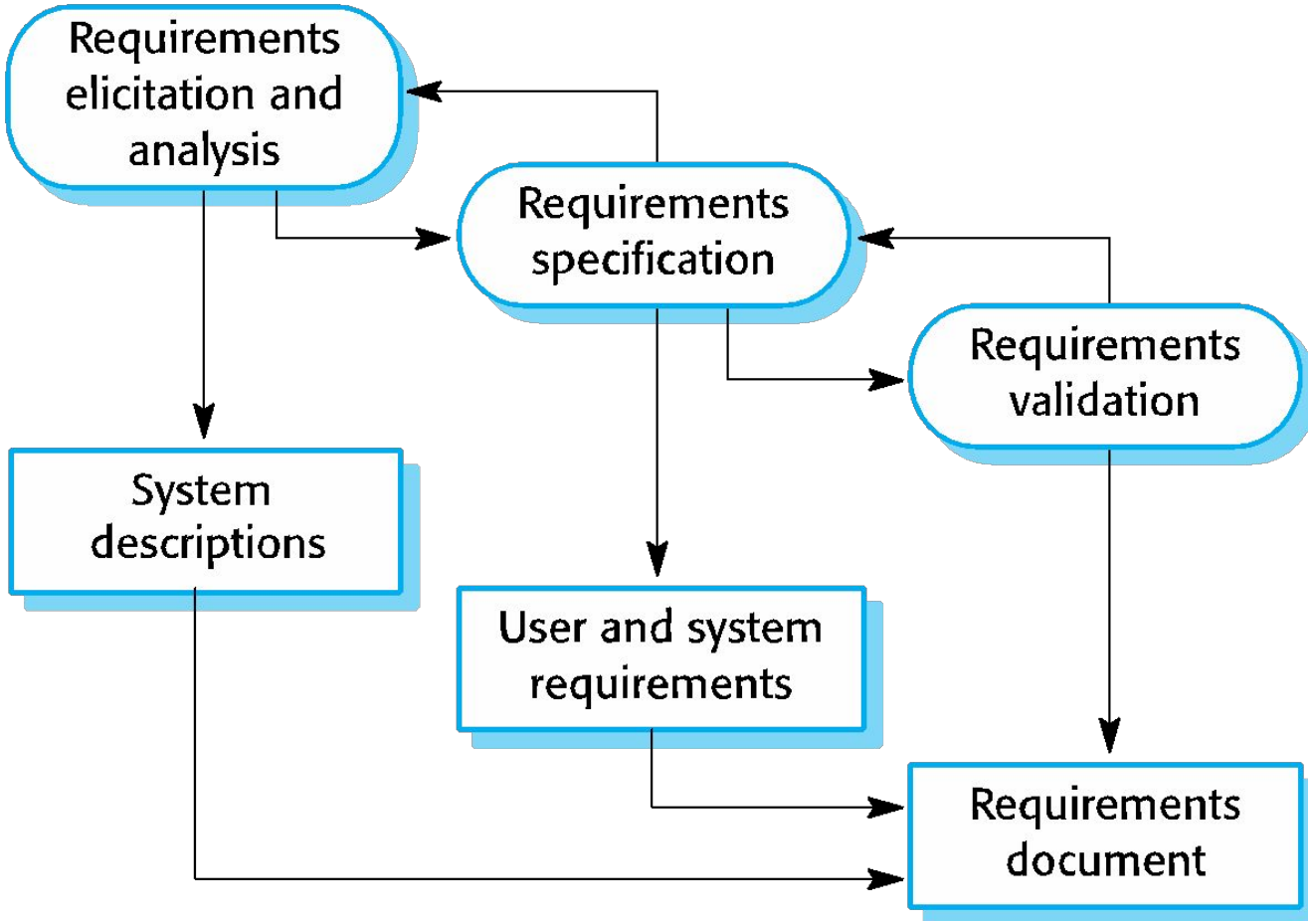
Types of reusable software

- Stand-alone application systems (sometimes called COTS) that are configured for use in a particular environment.
- Collections of objects that are developed as a package to be integrated with a component framework.
- Web services that are developed according to service standards and which are available for remote invocation.

Process activities

- Real software processes are inter-leaved sequences of technical, collaborative and managerial activities
  - Overall goal of specifying, designing, implementing and testing a software system.
- Four basic process activities
  - specification, development, validation and evolution
  - Organized differently in different development processes.
- For example, in the waterfall model, they are organized in sequence, whereas in incremental development they are interleaved.

The requirements engineering process



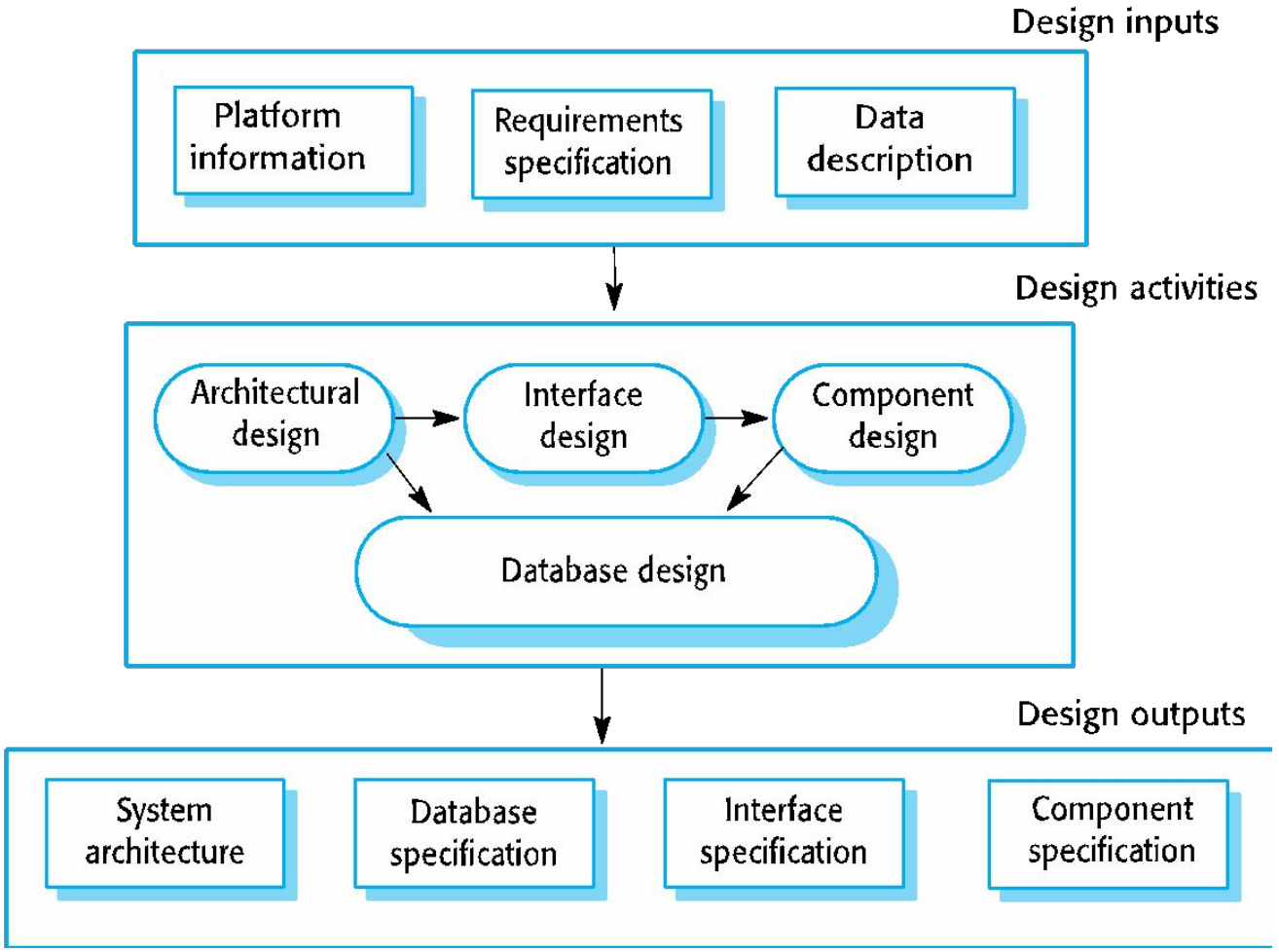
Software specification

- The process of establishing what services are required and the constraints on the system's operation and development.
- Requirements engineering process
  - 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

Software design and 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



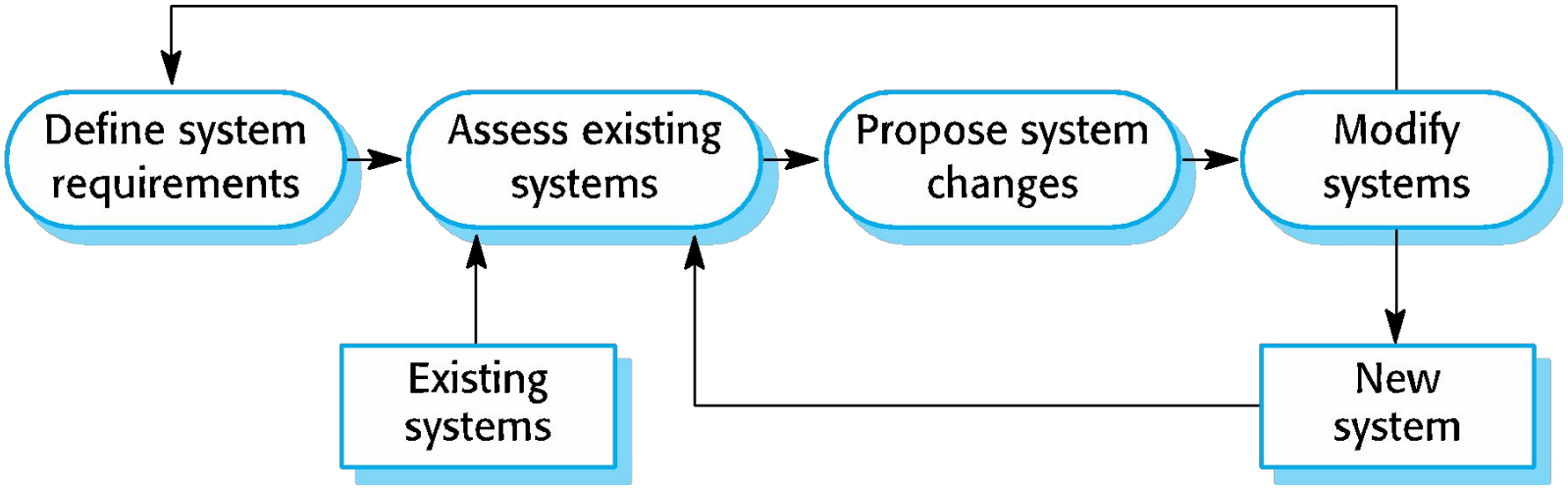
Software validation

- 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.

Software evolution

- Software is inherently flexible and can change.
- As requirements change so must software.
- Traditional distinction between development and evolution (maintenance) is increasingly irrelevant
  - Fewer and fewer systems are completely new.

System evolution



Coping with change

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 so the costs of change include both rework (e.g. re-analyzing requirements) as well as the costs of implementing new functionality

Reducing the costs of rework

- **Change anticipation**, 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.
- **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.

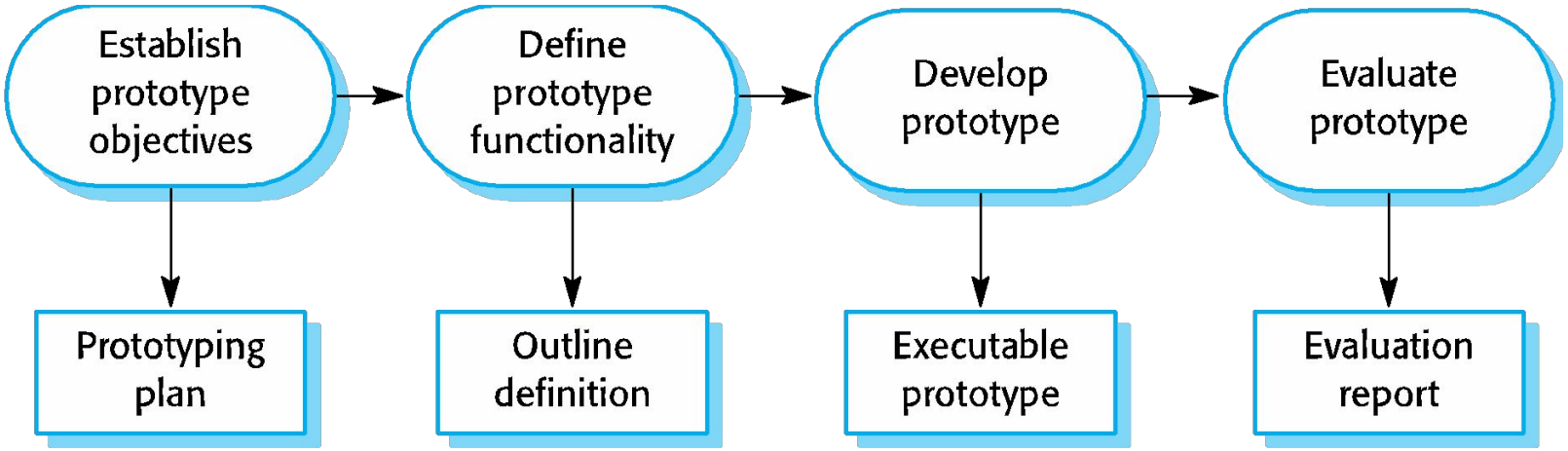
Coping with changing requirements

- **System prototyping**, where a version of the system or part of the system is developed quickly to check the customer's requirements and the feasibility of design decisions.
  - This approach supports change anticipation.
- **Incremental delivery**, where system increments are delivered to the customer for comment and experimentation.
  - This supports both change avoidance and change tolerance.

Software prototyping

- A prototype is an initial version of a system used to demonstrate concepts and try out design options.
  - May be thrown away.
- 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



Benefits of prototyping

- Improved system usability.
- A closer match to users' real needs.
- Improved design quality.
- Improved maintainability.
- Reduced development effort.

Incremental Delivery

- Early increments can be used as prototypes and gain experience for later system increments.
  - unlike prototypes, they are parts of the system
- Customers do not have to wait until the entire system is delivered
- Based on incremental development so easy to incorporate changes to systems.
- The highest priority services are delivered first and receive the most testing; therefore reduce the chance of software failiure.

Any problem of Incremental Delivery?

- When it is used for building a new system to replace an old system, what problem would cause for the the old system users?
- For government customers, system specification would be written into the contract at the very first step. In this case, is it possible to use incremental delivery?
- Any other kind of systems are not suitable to use incremental delivery?