SOFTWARE ENGINEERING

Spring 2024

SOFTWARE PROCESS

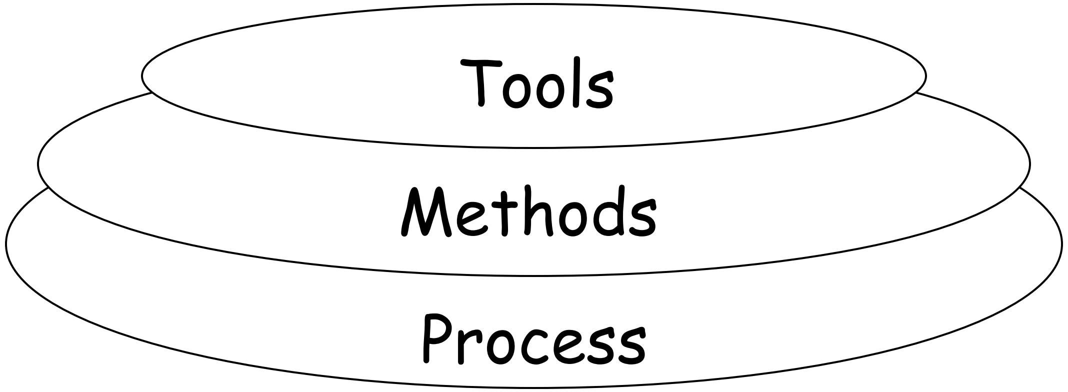
THE SOFTWARE PROCESS

- 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.
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

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 ENGINEERING LAYERS



- Process: framework of the required tasks
  - e.g., waterfall, extreme programming
- Methods: technical "how to"
  - e.g., design review, code review, testing
- Tools: automate processes and methods

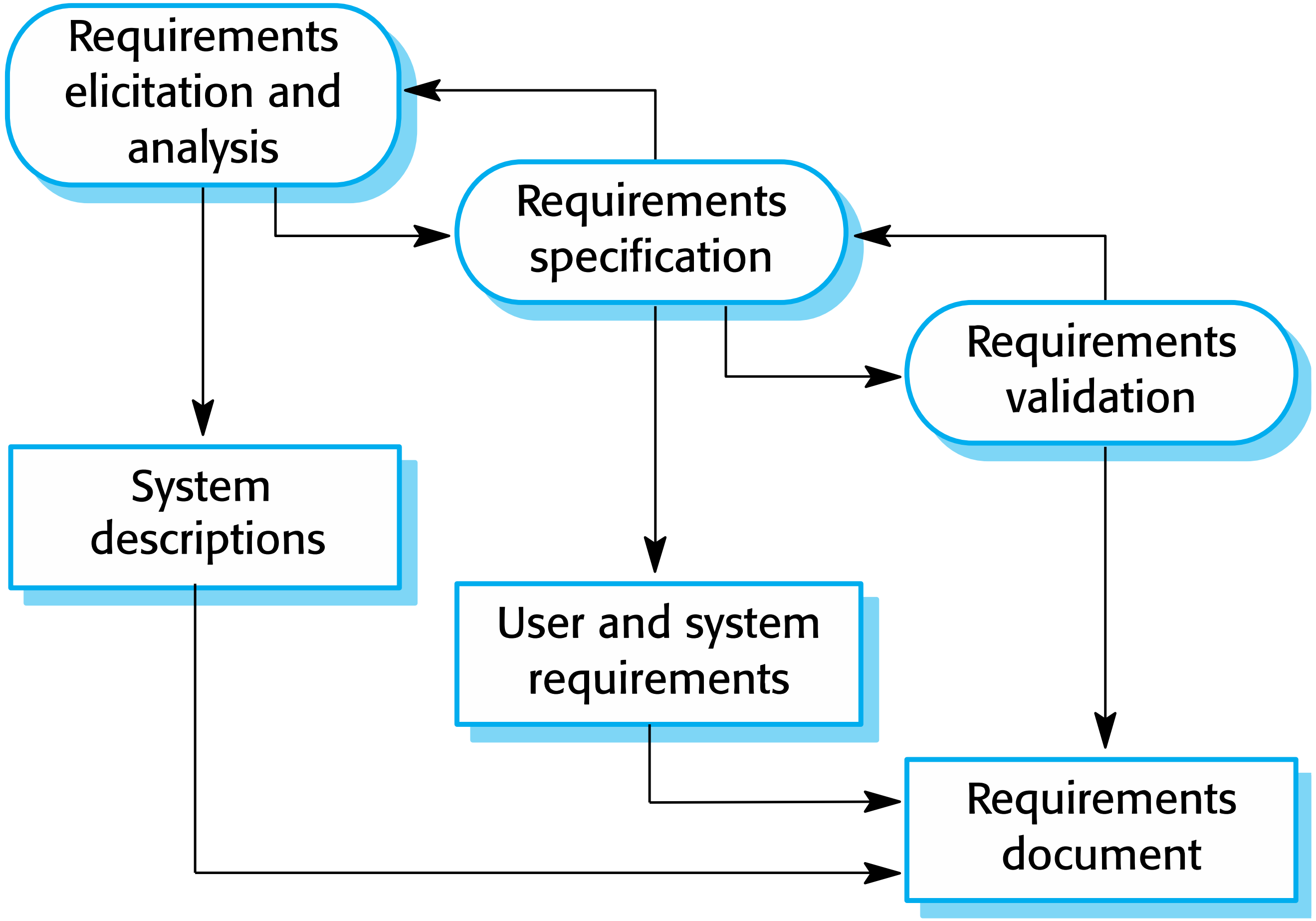
PROCESS ACTIVITIES

- 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 of specification, development, validation and evolution are organized differently in different development processes.
- For example, in the waterfall model, they are organized in sequence, whereas in incremental development they are interleaved.

1. SOFTWARE SPECIFICATION

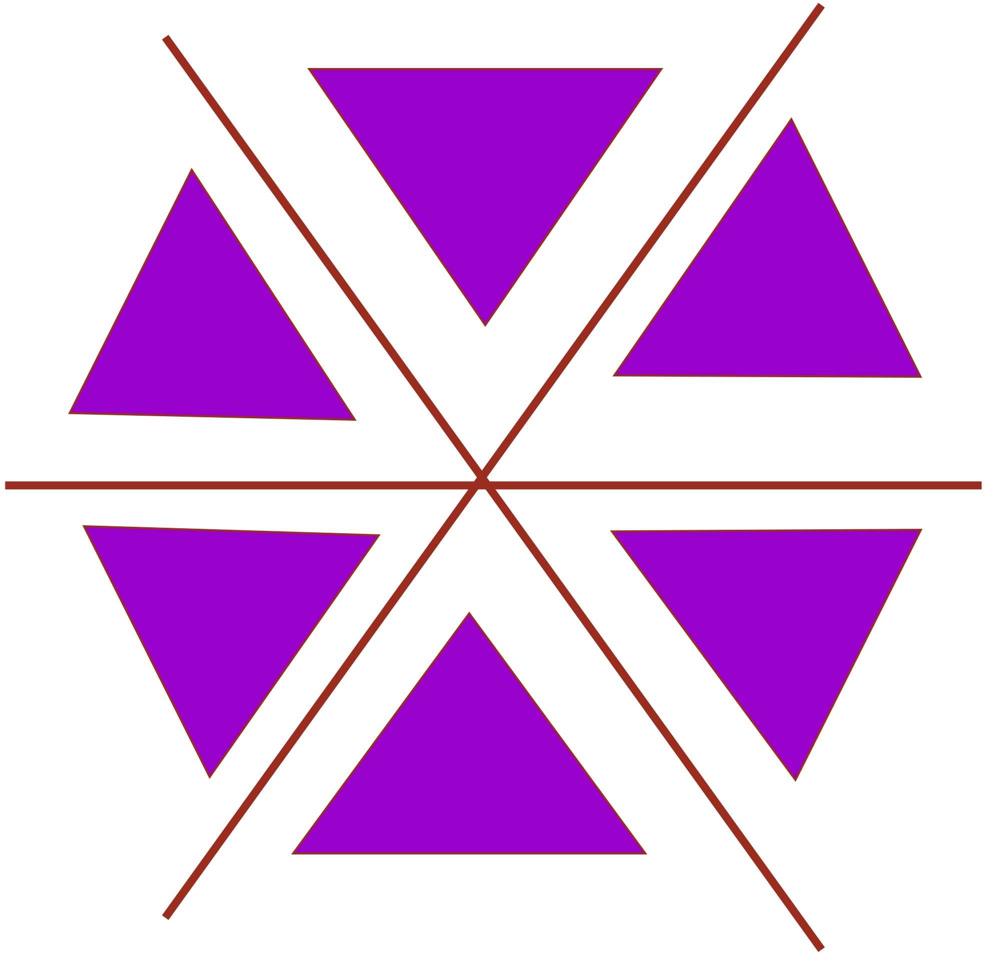
- The process of establishing what services are required and the constraints on the system’s operation and development.
- Requirements engineering process
  - Requirements elicitation and analysis
    - What do the system stakeholders require or expect from the system?
  - Requirements specification
    - Defining the requirements in detail
  - Requirements validation
    - Checking the validity of the requirements

THE REQUIREMENTS ENGINEERING PROCESS


Requirements elicitation and analysis
Requirements specification
Requirements validation
System descriptions
User and system requirements
Requirements document

2. SOFTWARE DESIGN AND IMPLEMENTATION

- The system architecture
- Decompose system in modules
- Specify interfaces between modules
- Much more of *how* the system works, rather than *what* it does

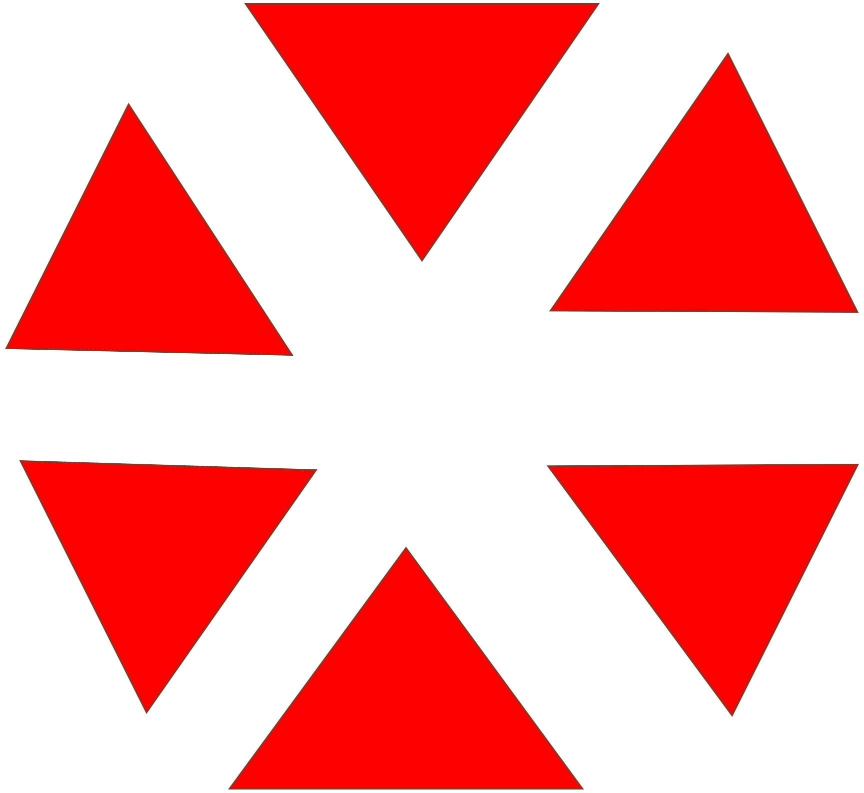


2. SOFTWARE DESIGN AND IMPLEMENTATION

- Code up the design

- First, make a plan
  - The order in which things will be done
  - Usually by priority
  - Also for testability

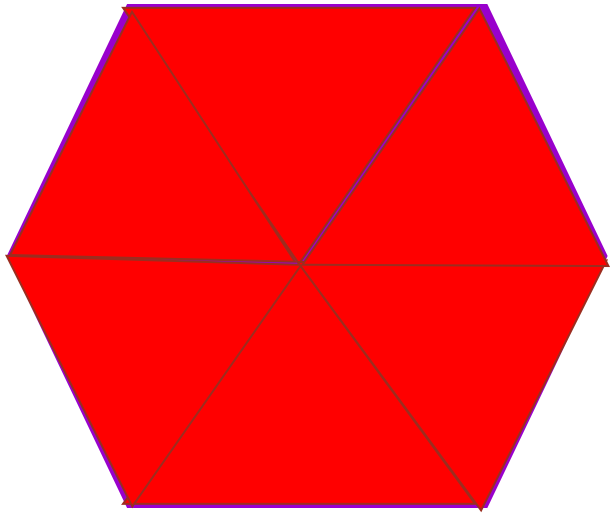
- Test each module



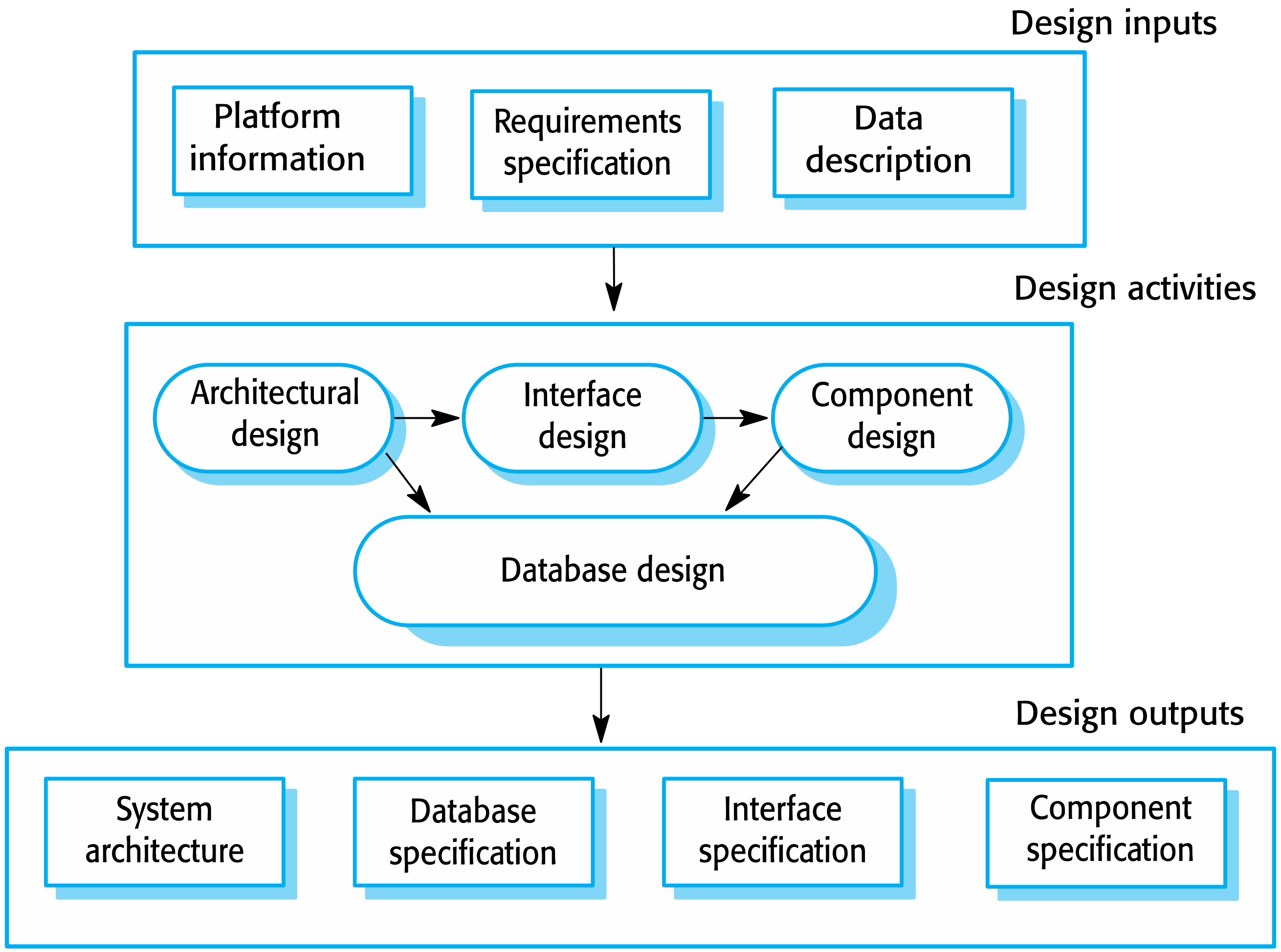
2. SOFTWARE DESIGN AND IMPLEMENTATION

- Put the pieces together

- A major QA effort at this point to test the entire system



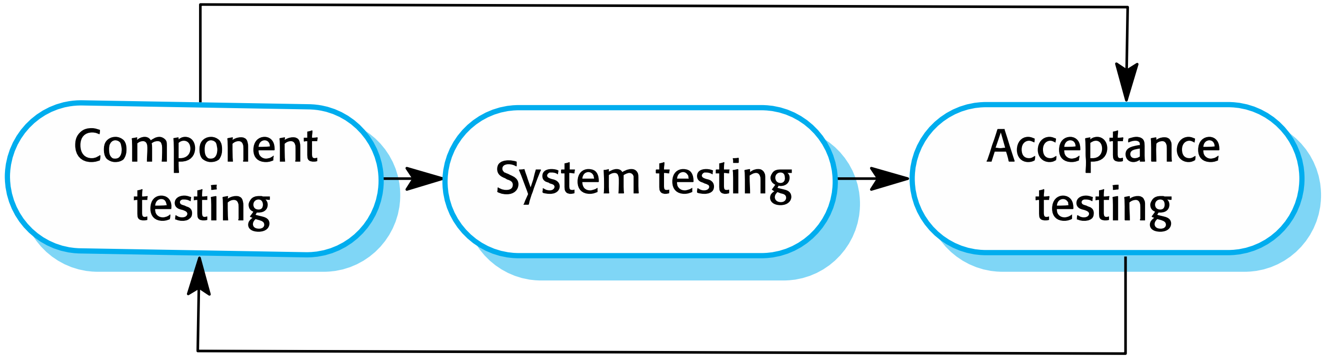
A GENERAL MODEL OF THE DESIGN PROCESS



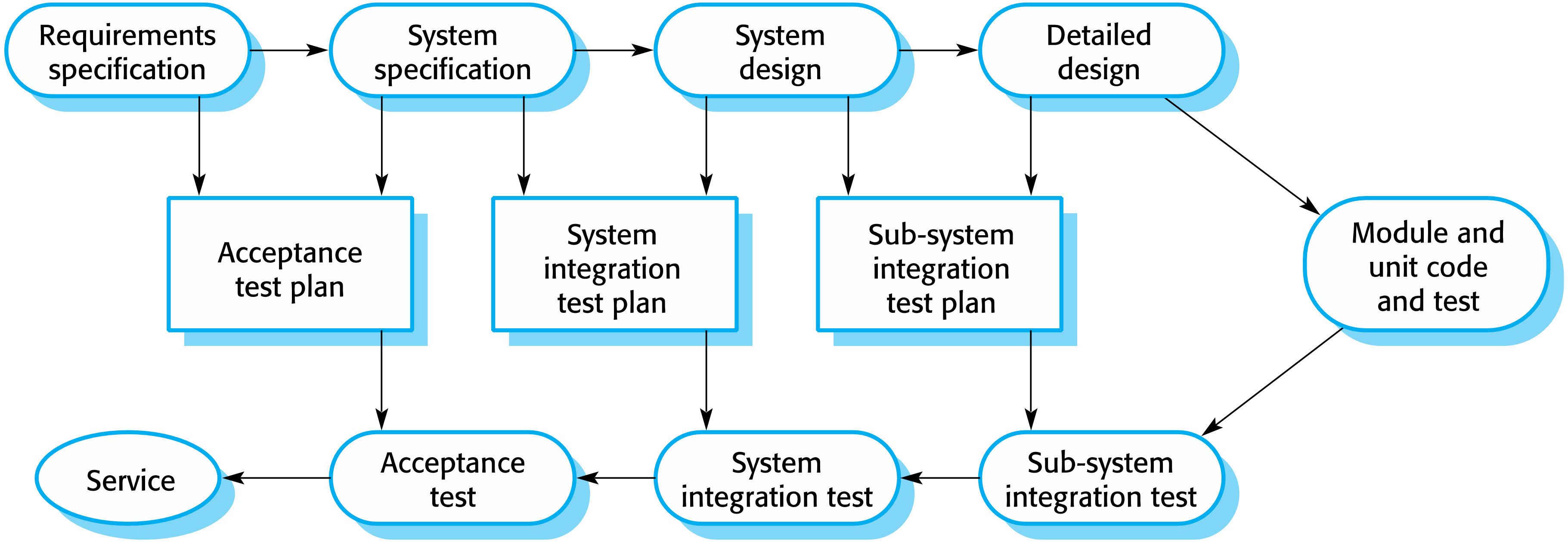
3. SOFTWARE VALIDATION

- 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



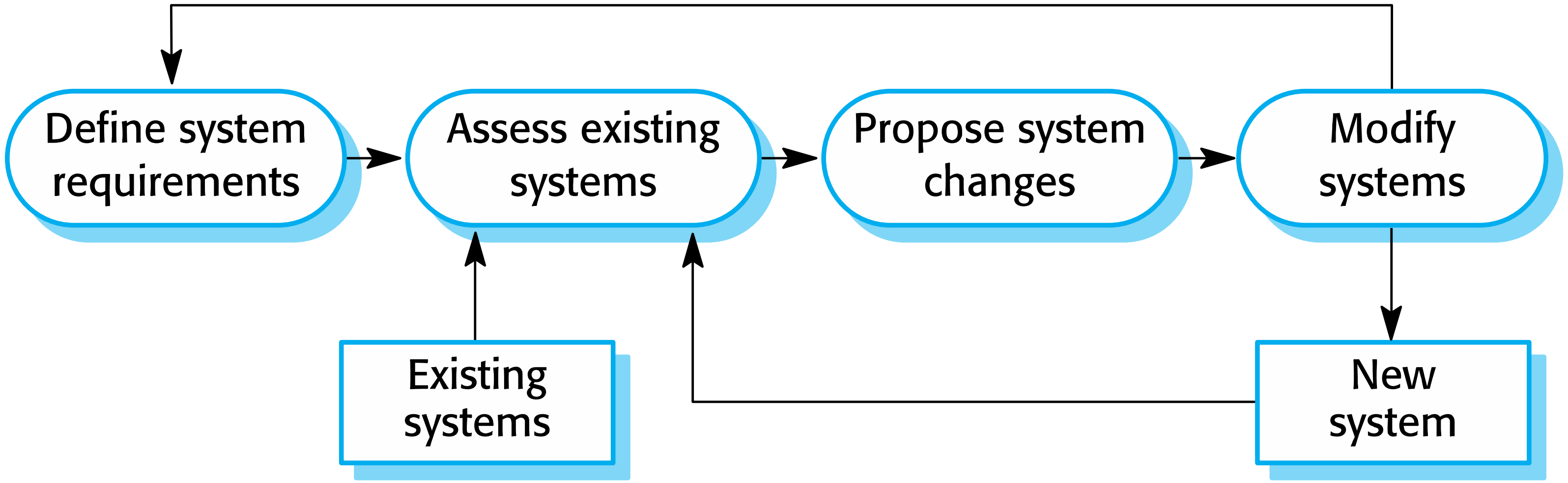
TESTING PHASES IN A PLAN-DRIVEN SOFTWARE PROCESS


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

4. 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.

SYSTEM EVOLUTION



SOFTWARE PROCESS MODELS

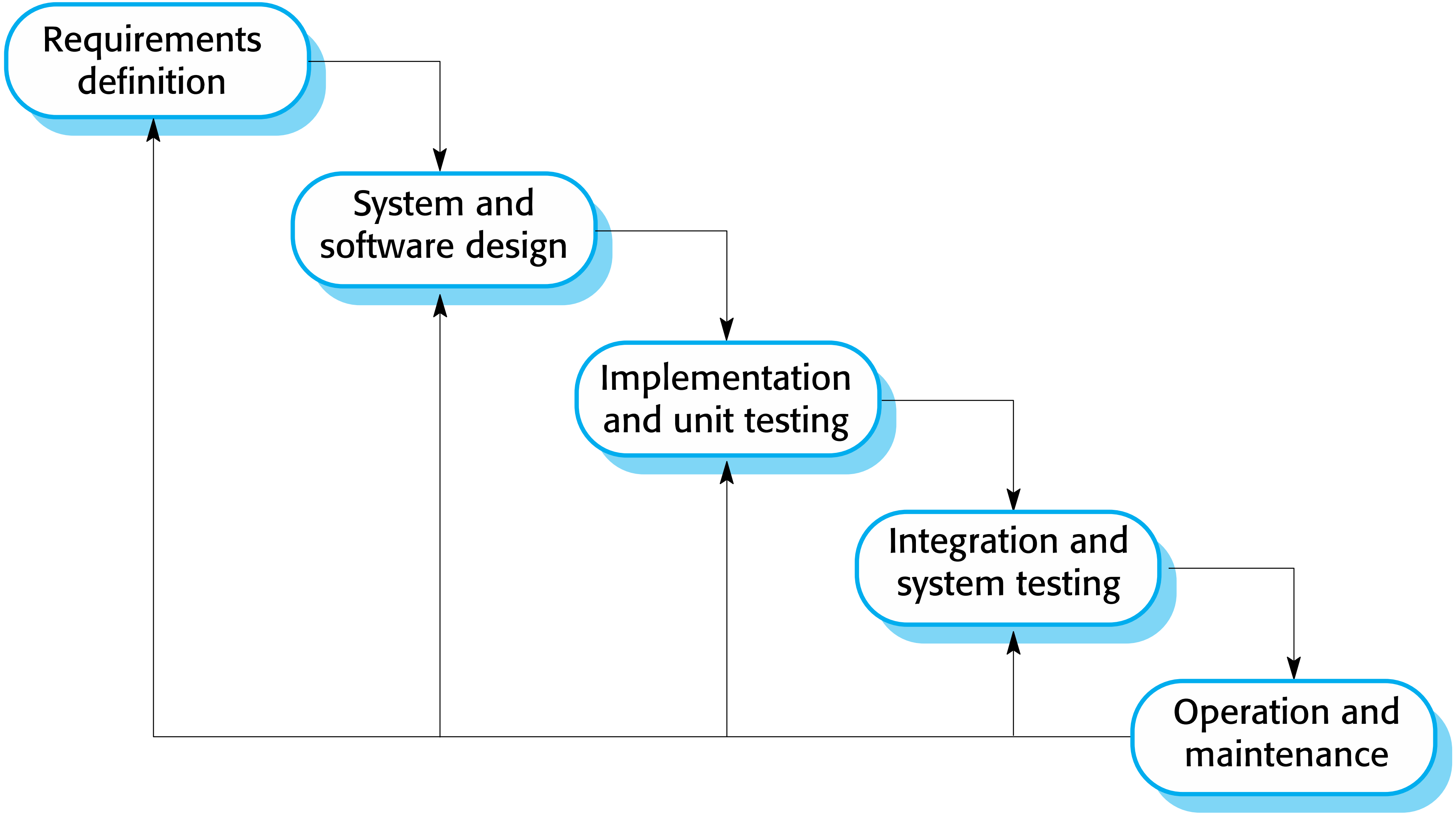
- The waterfall model
  - Plan-driven model. Separate and distinct phases of specification and development.
- Incremental development
  - Specification, development and validation are incorporated. May be plan-driven or agile.
- Integration and configuration
  - The system is assembled from existing configurable 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.

A SOFTWARE PROCESS: WATERFALL MODEL

- One of the standard models for developing software

- Each stage leads on to the next
  - No iteration or feedback between stages

WATERFALL PROCESS PHASES


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

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

OPINIONS

- The major risks are:
  - Relies heavily on being able to accurately assess requirements at the start
  - Little feedback from users until very late
    - Unless they understand specification documents
  - Problems in the specification may be found very late
    - Coding or integration
  - Whole process can take a long time before the first working version is seen
    - Frequent intermediate builds are needed to build confidence for a team
  - Sequential
- The programmers have nothing to do until the design is ready
- The waterfall model seems to be adopted from other fields of engineering
  - This is how to build bridges
- But many good aspects
  - Emphasis on spec, design, testing
  - Emphasis on communication through documents

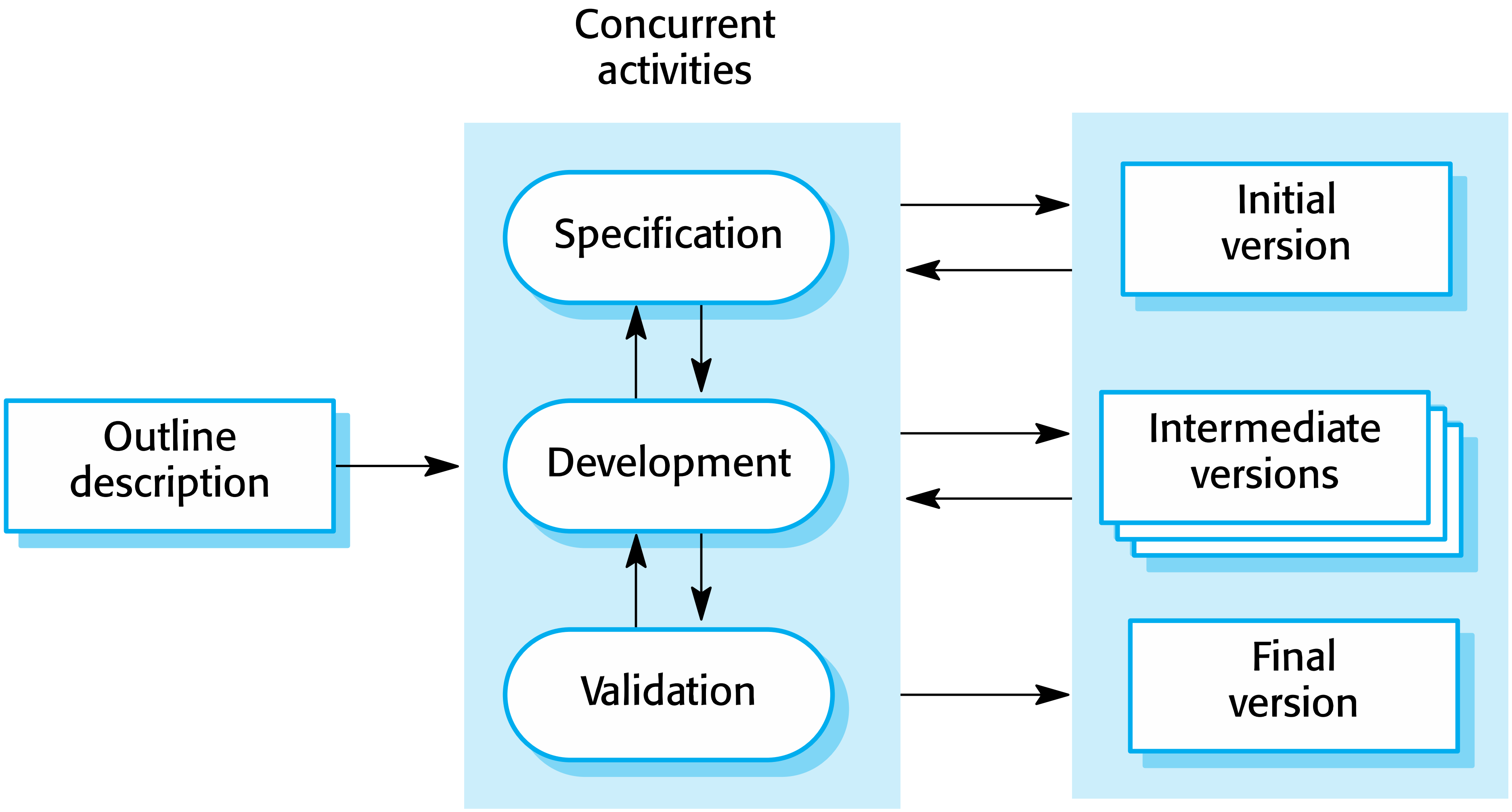
AN OPINION ON TIME

- Time is the enemy of all software projects

- Taking a long time is inherently risky

"It is hard to make predictions,
especially about the future"

INCREMENTAL DEVELOPMENT



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.

INCREMENTAL DEVELOPMENT PROBLEMS

- The process is not visible.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

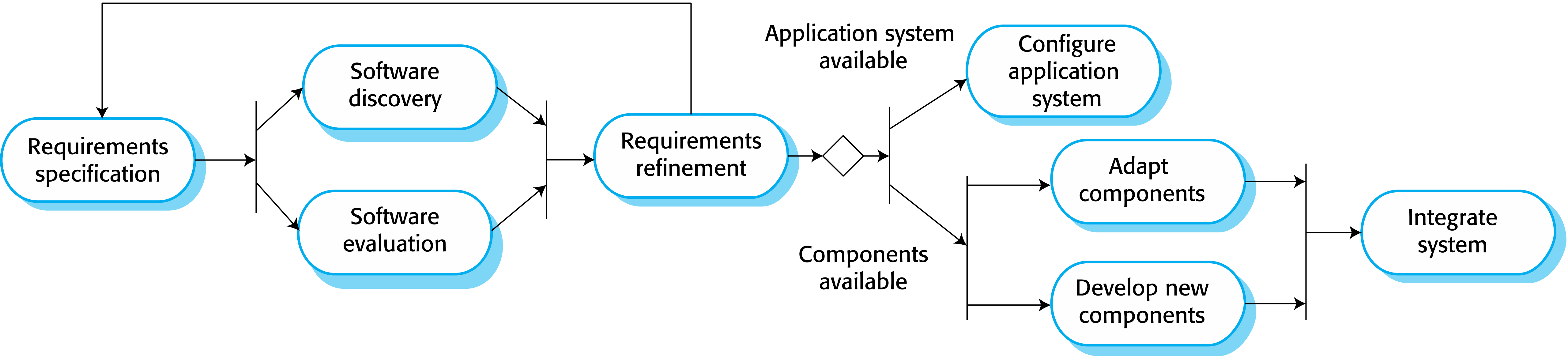
INTEGRATION AND CONFIGURATION

- Based on software reuse where systems are integrated from existing components or application systems (sometimes called COTS -Commercial-off-the-shelf) 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

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 such as .NET or J2EE.
- Web services that are developed according to service standards and which are available for remote invocation.

REUSE-ORIENTED SOFTWARE ENGINEERING



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.

SOMETHING FASTER: RAPID PROTOTYPING

- Write a quick prototype

- Show it to users
  - Use to refine requirements

- Then proceed as in waterfall model
  - Throw away the prototype
  - Do spec, design, coding, integration, etc.

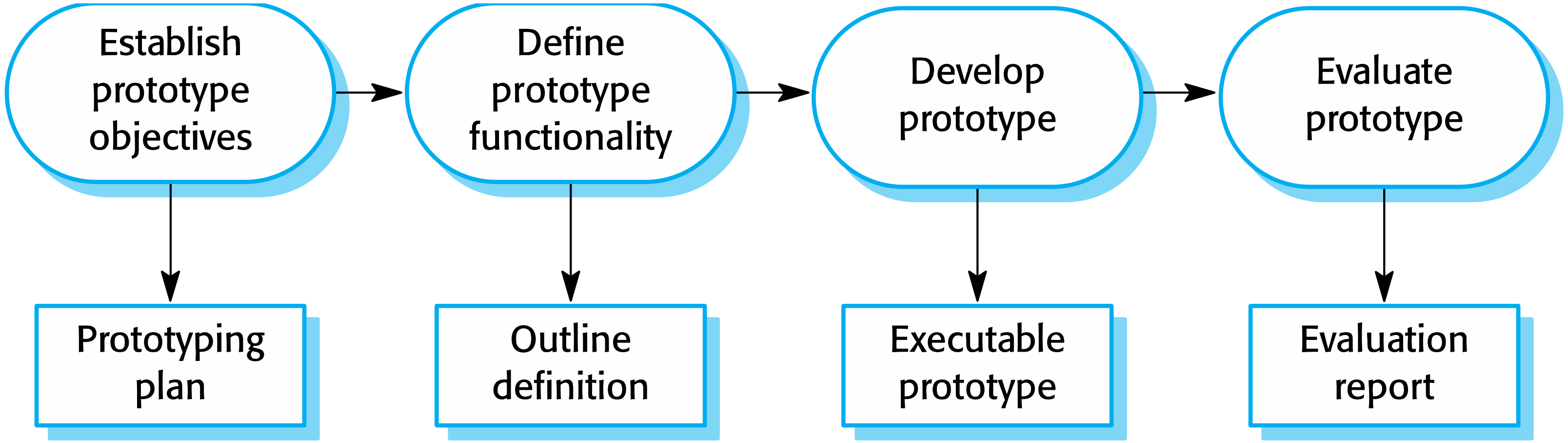
COMMENTS ON RAPID PROTOTYPING

- Hard to throw away the prototype
  - Slogan "the prototype is the product"
  - Happens more often than you might think!
- A prototype is useful in refining requirements
  - Much more realistic to show users a system rather than specification documents
- A prototype exposes design mistakes
- Experience building a prototype will improve greatly the accuracy of plans

OPINIONS ON REALITY

- Neither of these models is true to life
- In reality, feedback between all stages
  - Specifications will demand refined requirements
  - Design can affect the specification
  - Coding problems can affect the design
  - Final product may lead to changes in requirements
    - I.e., the initial requirements weren’t right!

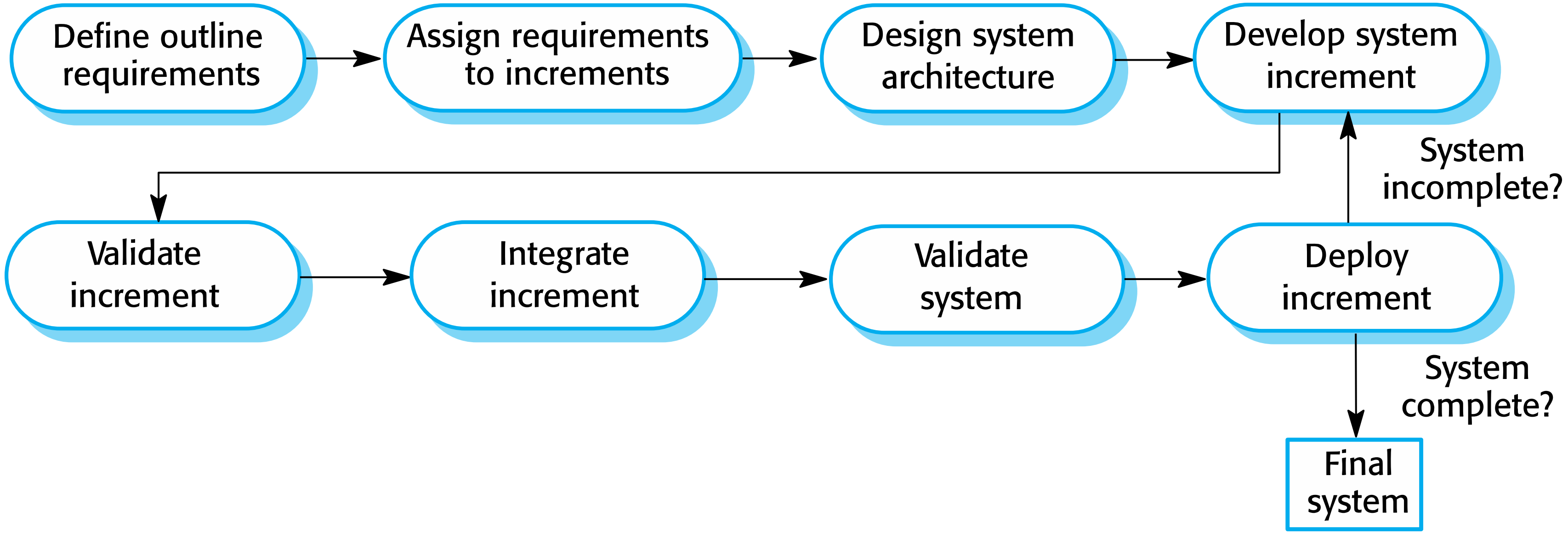
THE PROCESS OF PROTOTYPE DEVELOPMENT



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



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.
- Lower risk of overall project failure.
- The highest priority system services tend to receive the most testing.

INCREMENTAL DELIVERY PROBLEMS

- Most systems require a set of basic facilities that are used by different parts of the system.
    - As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.
- The essence of iterative processes is that the specification is developed in conjunction with the software.
    - However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.

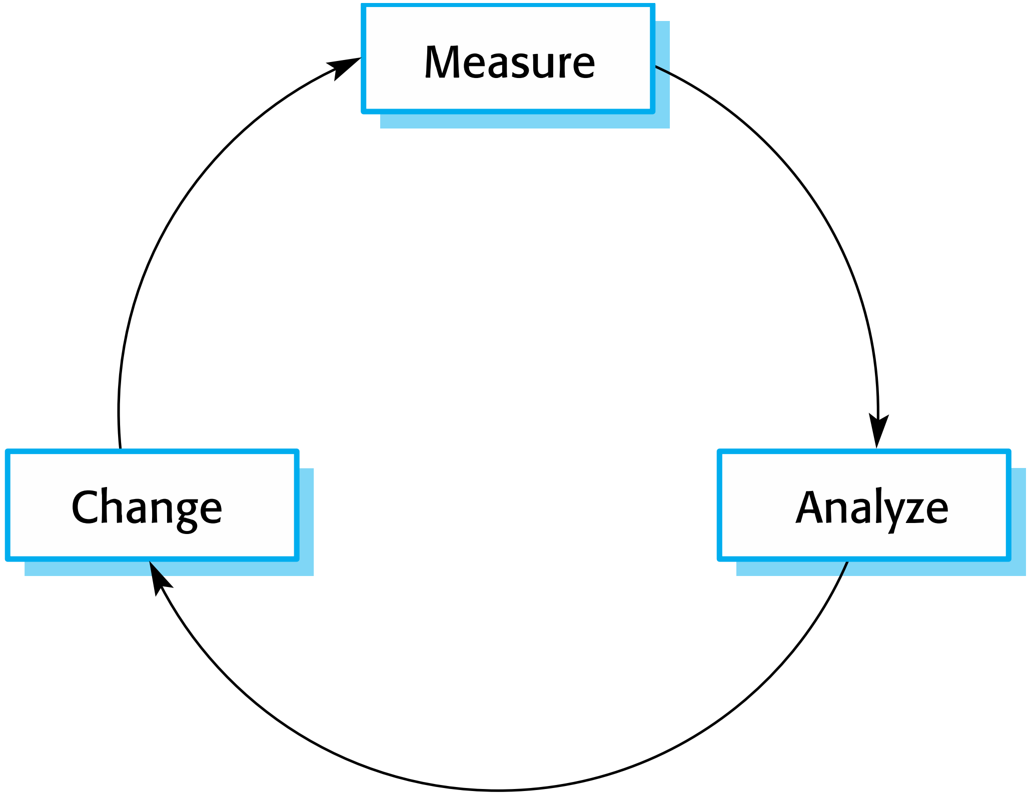
PROCESS IMPROVEMENT

- Many software companies have turned to software process improvement as a way of enhancing the quality of their software, reducing costs or accelerating their development processes.
- Process improvement means understanding existing processes and changing these processes to increase product quality and/or reduce costs and development time.

APPROACHES TO IMPROVEMENT

- The process maturity approach, which focuses on improving process and project management and introducing good software engineering practice.
  - The level of process maturity reflects the extent to which good technical and management practice has been adopted in organizational software development processes.
- The agile approach, which focuses on iterative development and the reduction of overheads in the software process.
  - The primary characteristics of agile methods are rapid delivery of functionality and responsiveness to changing customer requirements.

THE PROCESS IMPROVEMENT CYCLE



PROCESS IMPROVEMENT ACTIVITIES

- *Process measurement*
  - You measure one or more attributes of the software process or product. These measurements forms a baseline that helps you decide if process improvements have been effective.
- *Process analysis*
  - The current process is assessed, and process weaknesses and bottlenecks are identified. Process models (sometimes called process maps) that describe the process may be developed.
- *Process change*
  - Process changes are proposed to address some of the identified process weaknesses. These are introduced and the cycle resumes to collect data about the effectiveness of the changes.

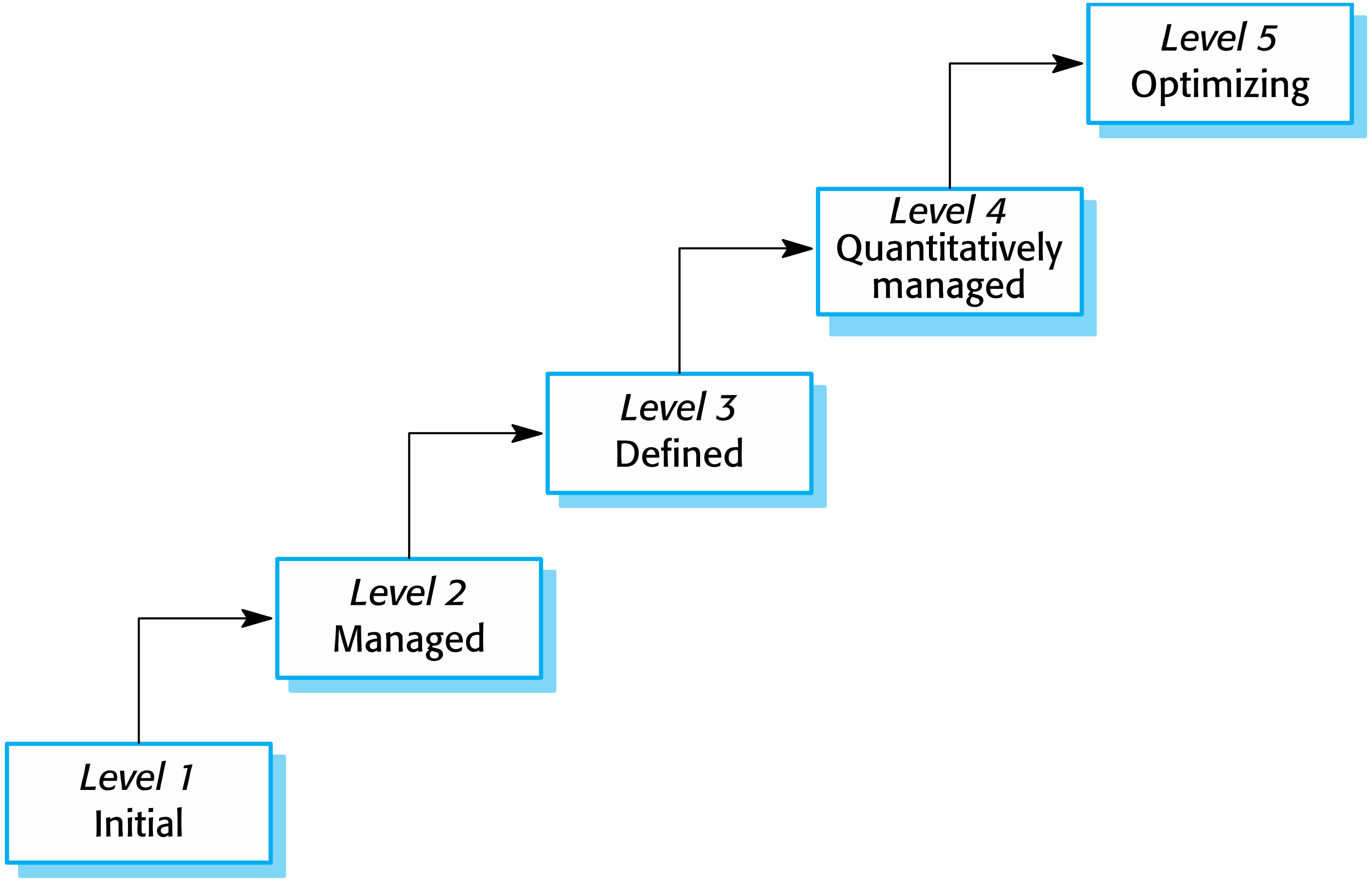
PROCESS MEASUREMENT

- Wherever possible, quantitative process data should be collected
  - However, where organisations do not have clearly defined process standards this is very difficult as you don't know what to measure. A process may have to be defined before any measurement is possible.
- Process measurements should be used to assess process improvements
  - But this does not mean that measurements should drive the improvements. The improvement driver should be the organizational objectives.

PROCESS METRICS

- Time taken for process activities to be completed
  - E.g. Calendar time or effort to complete an activity or process.
- Resources required for processes or activities
  - E.g. Total effort in person-days.
- Number of occurrences of a particular event
  - E.g. Number of defects discovered.

CAPABILITY MATURITY LEVELS



THE SEI CAPABILITY MATURITY MODEL

- Initial
  - Essentially uncontrolled
- Repeatable
  - Product management procedures defined and used
- Defined
  - Process management procedures and strategies defined and used
- Managed
  - Quality management strategies defined and used
- Optimising
  - Process improvement strategies defined and used