Software Engineering
CS-303

Software Process & Process Models

Lecture # 4, 5, 6
28, 29, 30 Jan

Rubab Jaffar
rubab.jaffar@nu.edu.pk

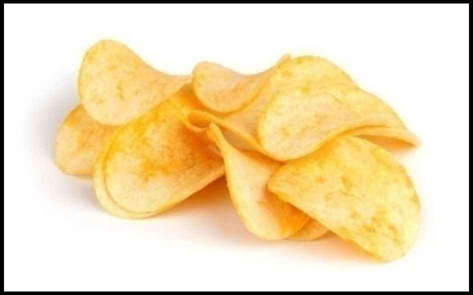
Today's Outline

- What is Process?
- Software Process
- Software Engineering Frame Work Process
- Process flow
- Software Process Models

Process

Process: A particular method, generally involving a number of steps.

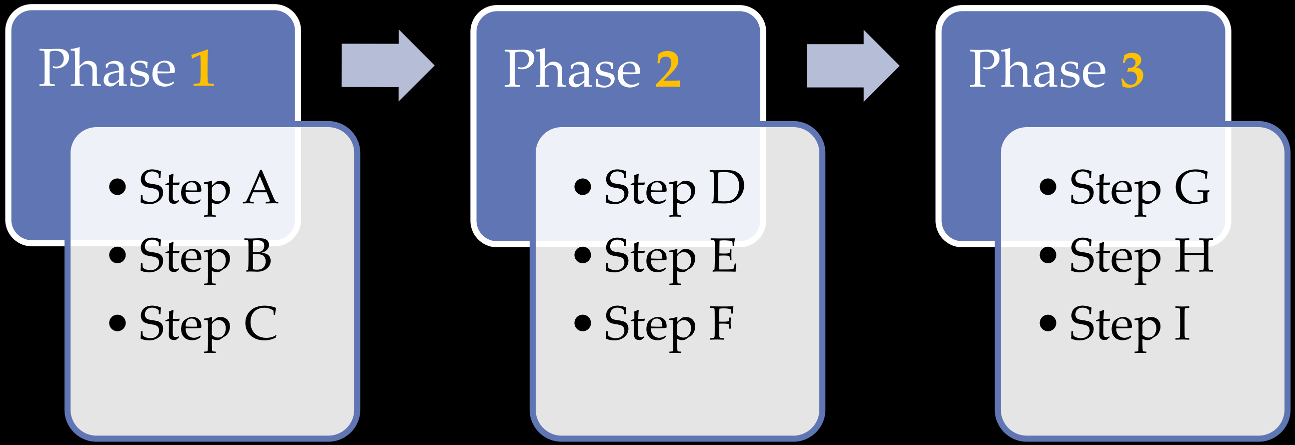
Process for making *potato chips*



Washing → Peeling → Frying → Flavoring → Packaging

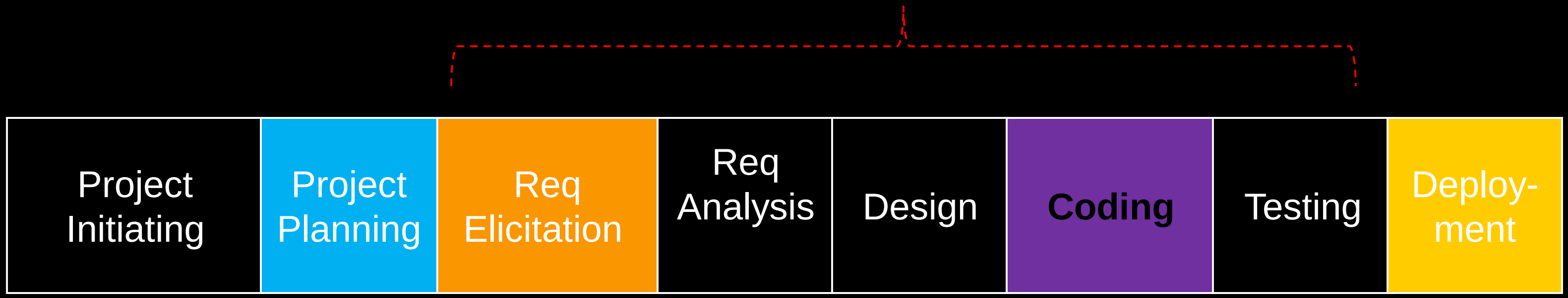
Process

- Process is generally a set of ***phases***.
- Each phase performs a well ***defined task*** and generally produces an output.



Software Process

- ***Software Process***: A set of steps, along with ordering constraints on execution, to produce software with desired outcome.
  - Many types of ***activities*** performed by ***different teams***
  - Software process is comprising of many component processes



Process

What we should do, to develop a *good quality* software ?

What are the attributes of a ***good process*** ?
Visible, Repeatable, Measureable

Can you give examples of ***Technical* and *Managerial***
problems in software development process ?

Project fails due to **Managerial** problems

Product

- ***Products*** are outcomes of executing a ***process*** for a project.
- Does Process quality and Product quality has any relation ?
- Software development life cycle (SDLC), is a structure imposed on the development of a software product.
- Software Engineering says if you follow the process the output is predicable and repeatable no matter who does that.
- Software Engineering focuses on ***process***.
- Mature processes will help achieve ***project objective***s of high Quality Product.

Software Process

- Although there is no ***ideal*** software process.
- There is scope for improving the software process in many organizations.
- Processes may include *outdated* techniques or may not take advantage of the best practice in software engineering.
- Software processes can be improved by process *standardization* where the diversity in software processes across an organization is reduced.

Software Engineering Framework

Software Engineering Framework

- What is ***framework*** and why we need framework?
  - o Framework means; set of rules to be followed.

- What are ***those rules***? Those rules have been ***adopted*** by organizations that produce good results.

- Experts convert those rules into a framework to be used by every organization with respect to their needs.

- **Example:** Framework for Requirements Development

Software Engineering Framework

What

Definition

How

Development

Change

Maintenance

Umbrella / parallel activities

- Quality Assurance
- Configuration Mg'mt
- Project Monitoring
- Measurement

Definition Phase

- *Definition* phase focuses on *what* (is required).

- During definition, <u>SW-development-Team</u> and <u>user</u> attempts to identify the following questions:
  - o What is ***need*** (or problem)?
  - o What *features* are required?
  - o What interfaces are to be established?
  - o Any budget or technical ***constraints*** ?
  - o What is *success criteria* ?.

Development Phase

- *Development* phase focuses on the **how**.

- During development, the SW-development-Team attempts to define how:
  - How database would be designed
  - How software *architectures* would be designed
  - How the design will translate into programming language
  - How *testing* will be performed

Development Phase

- Methods applied during development phase, *will vary* (depending on the SDLC) but the three steps will occur in some form:

- *Design:* Design translate the requirements into some graphical or tabular representations.

- *Coding:* Design is then translated into programming language.

- *Testing:* The executable code must be tested to uncover errors.

Maintenance Phase

- Maintenance phase focuses on changes that associated with
  - Error Correction (Corrective)
  - Platform Adaptations required (Adaptive)
  - Enhancement due to change (Perfective)
  - The work carried out order to avoid any breakdown or malfunction (Preventive)

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.

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.

Today’s Outline

- Software process models
  - Water Fall Model
  - Evolutionary Development
  - Component base Software Engineering / Reuse-oriented development
- Process iteration
  - Incremental Model
  - Spiral Model
- Process activities

Software Process Model

- ***Software Process*** : is coherent sets of activities for specifying, designing, implementing and testing software systems.
- ***A software process model*** is an abstract representation of a software process.
- *It is a description of the sequence of activities carried out in an SE project, and the relative order of these activities.* It presents a description of a process from some particular perspective.

Generic software process models

- The waterfall model
  - Separate and distinct phases of specification and development. It is the oldest paradigm for SE. When requirements are well defined and reasonably stable, it leads to a linear fashion.

- Incremental development
  - Specification, development and validation are interleaved.

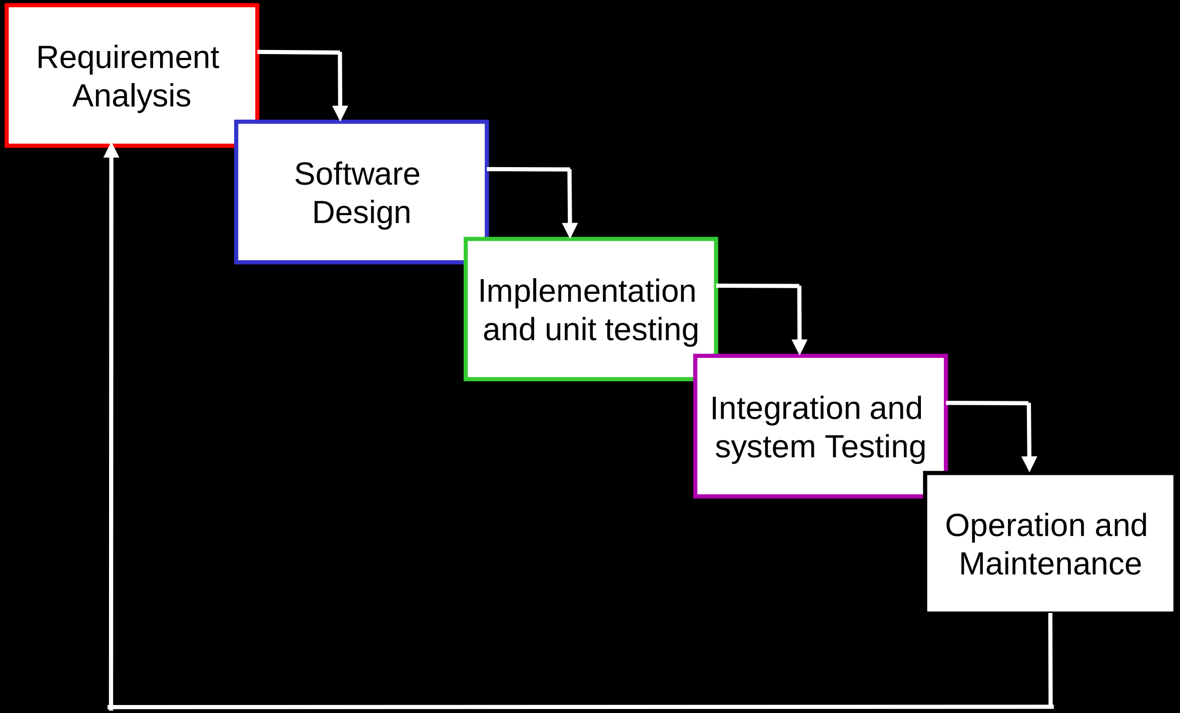
- Integration and configuration
  - The system is assembled from existing configurable components. May be plan-driven or agile.

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

Waterfall

Each box represents a **set of tasks** that results in the production of each or more work products.

Each new phase begins when the **work products** of the previous phase as completed, frozen and signed off.


Requirement Analysis
Software Design
Implementation and unit testing
Integration and system Testing
Operation and Maintenance

Waterfall (when to use)

- Well suited for projects where requirements can be understood easily and technology decisions are easy
- Has been used widely
- For standard/familiar type of projects it still may be the most optimum.

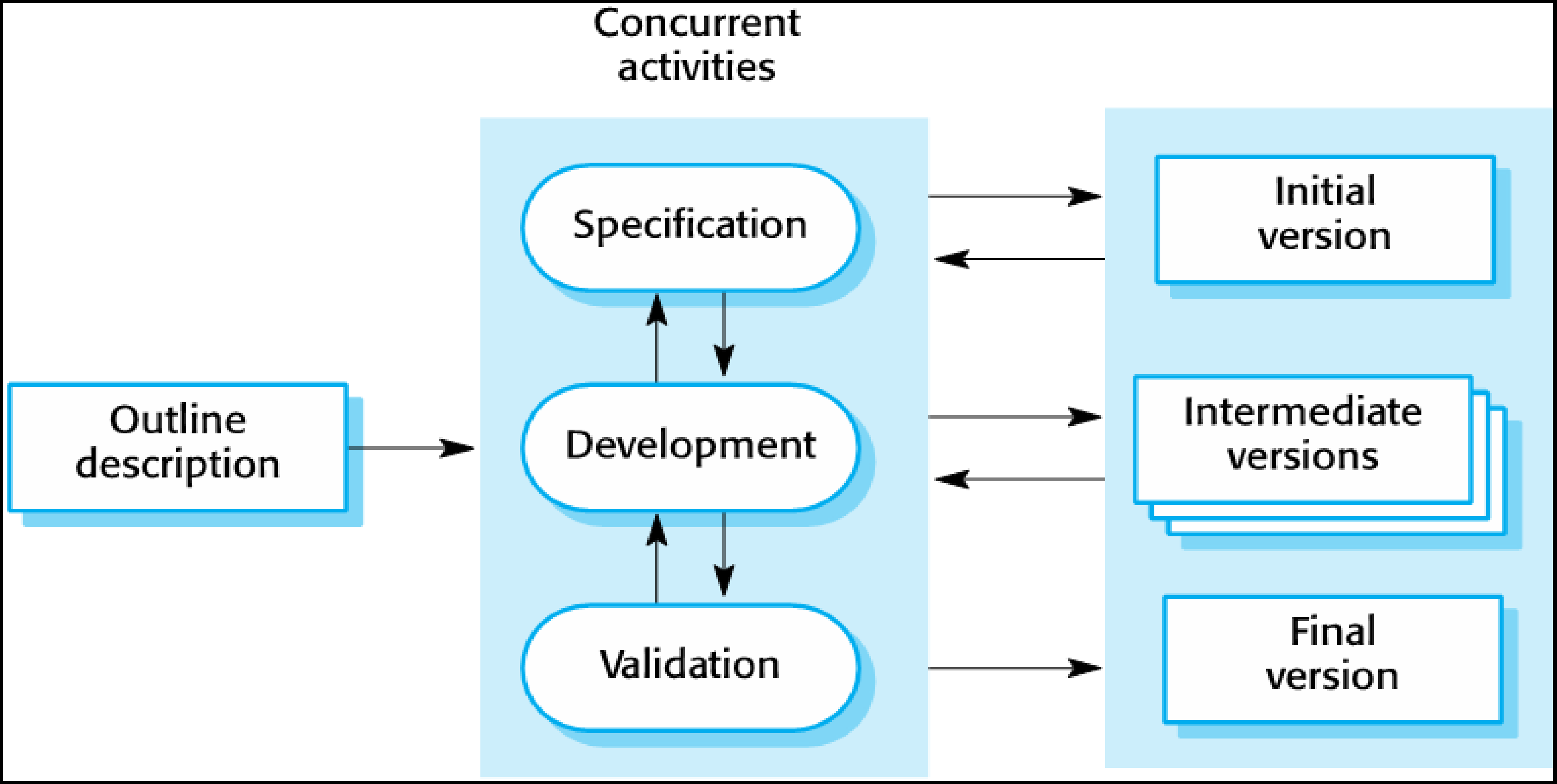
Waterfall model problems

- 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.

Disadvantages of Waterfall Model

- Unable to work where high level of uncertainty is involved
- Requirements need to be stable hence making model rigid
- Unrealistic to state all requirements at the beginning
- Does not handle concurrent events – development teams are delayed waiting for others
- Difficult and expensive to change decisions
- The user is involved only in the beginning phase of requirement gathering and than during acceptance phase

Iterative Development



Iterative 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.

Iterative 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.

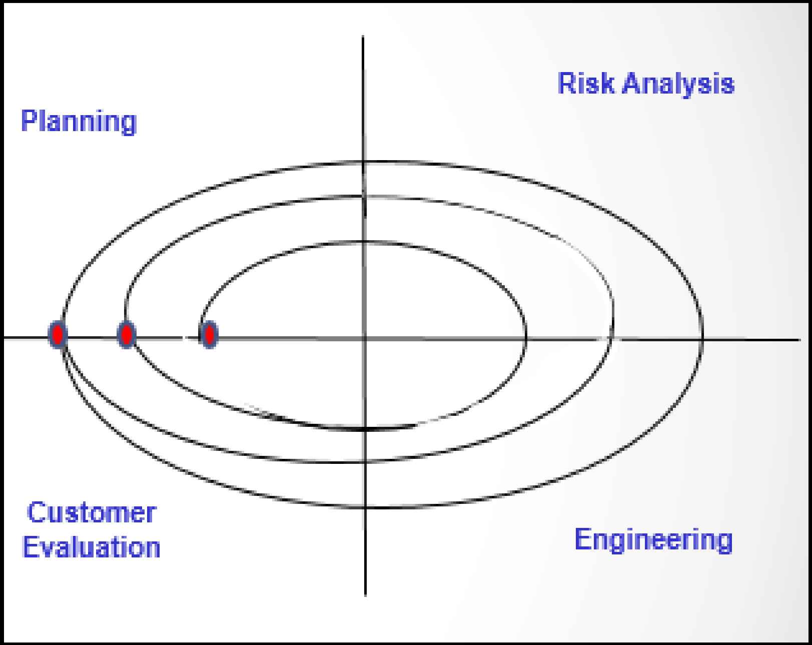
Evolutionary Models: Spiral

- Spiral is primary **Risk Driven** approach. Spiral model consist of different **cycle. In each cycle we try to address some risk elements.**

- Planning: Determines objectives, alternatives and constraints.
- Risk Analysis: Analysis of alternatives as well as an identification and/or resolution of risks.
- Engineering: Development of the next level of product
- Customer evaluation: Assessment of the results of engineering

Spiral

- With each iteration around the spiral progressively more complete versions of the software are built.
- Spiral model enables the developer, and the customer, to understand and ***react to risk*** at each evolutionary level.
- Each loop around the spiral implies that project costs and schedules may be modified.



- This creates problems in fixed-price project.

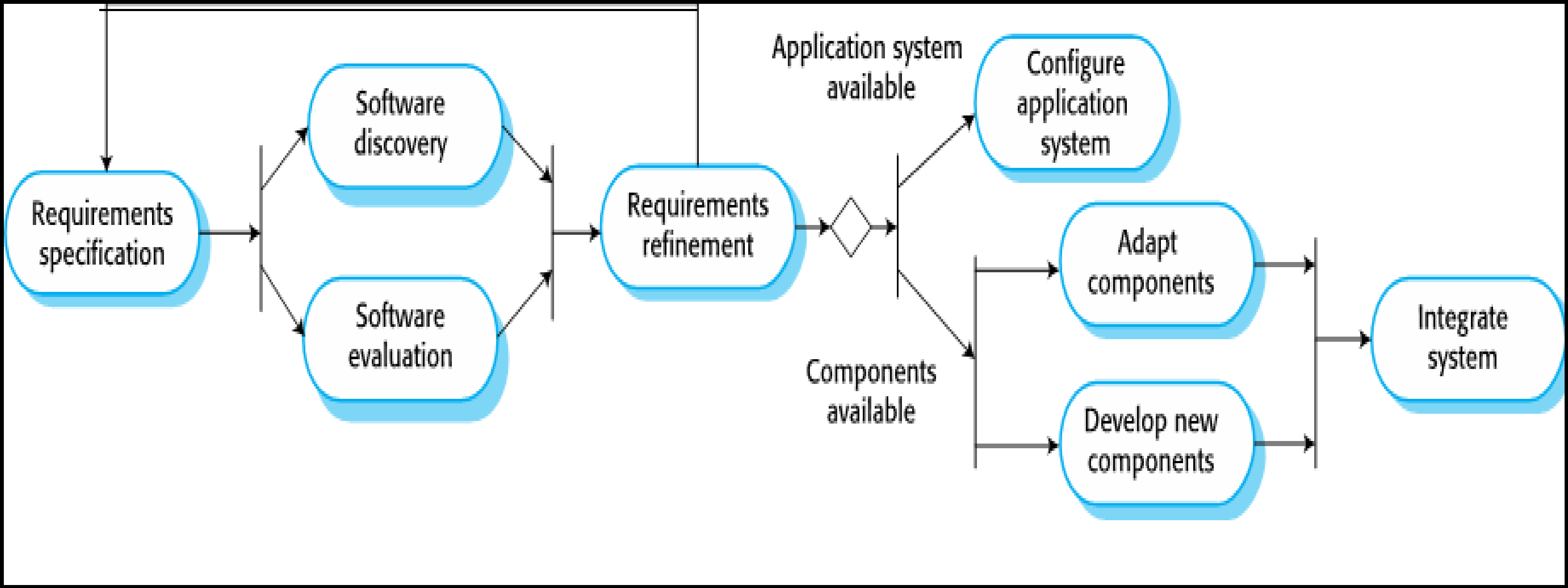
Integration and Configuration

- 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
  - 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 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



Key Process Stages

- Requirements specification
- Software discovery and evaluation
- Requirements refinement
- Application system configuration
- Component adaptation and integration

Advantages and Disadvantages

- Reduced costs and risks as less software is developed from scratch
- Faster delivery and deployment of system
- But requirements compromises are inevitable so system may not meet real needs of users
- Loss of control over evolution of reused system elements

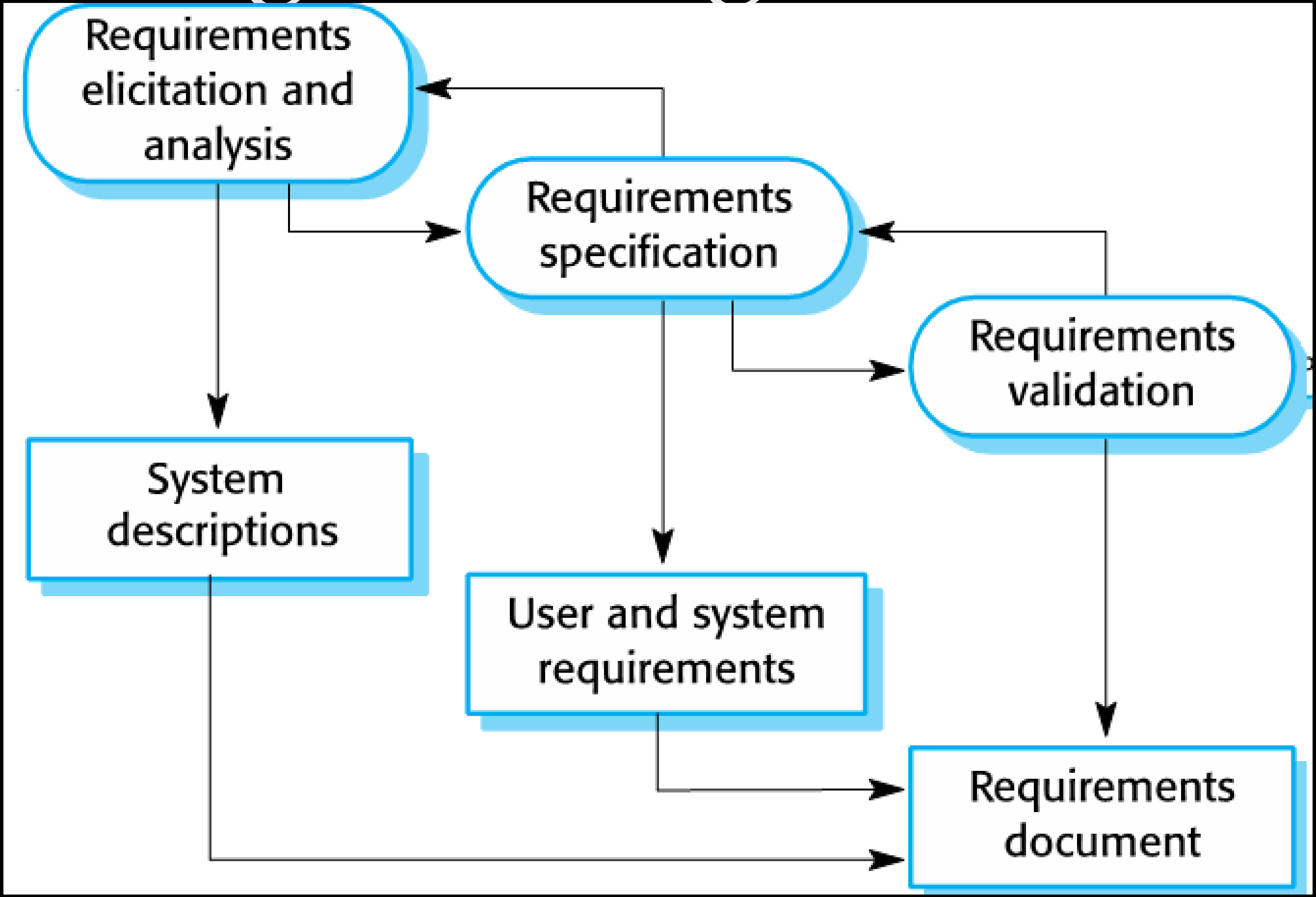
Process Activities

- 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 iterative development they are interleaved.

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

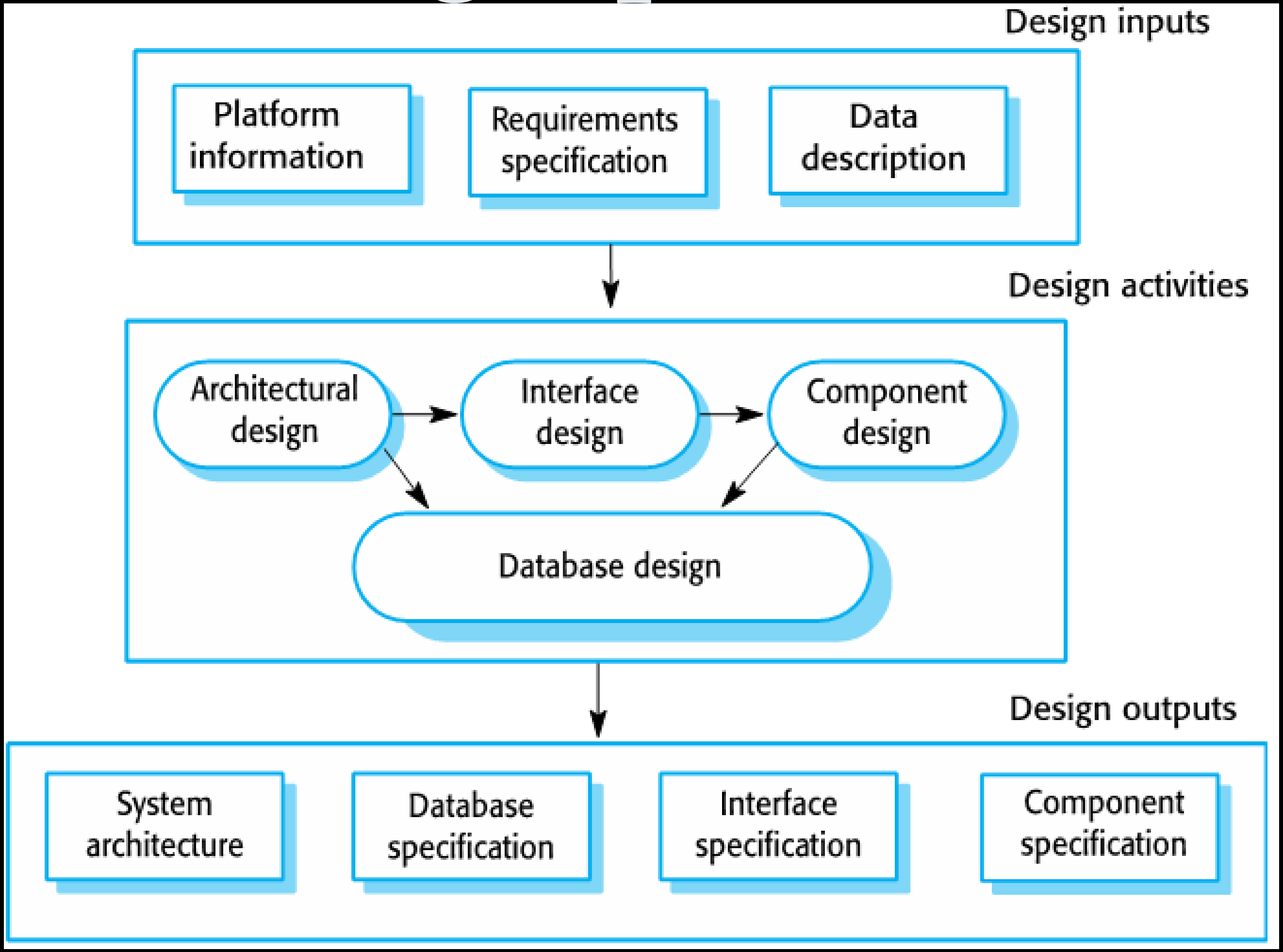
The Requirements Engineering Process


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

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



Design Activities

- *Architectural design,* where you identify the overall structure of the system, the principal components (subsystems or modules), their relationships and how they are distributed.
- *Database design,* where you design the system data structures and how these are to be represented in a database.
- *Interface design,* where you define the interfaces between system components.
- *Component selection and design,* where you search for reusable components. If unavailable, you design how it will operate.

System Implementation

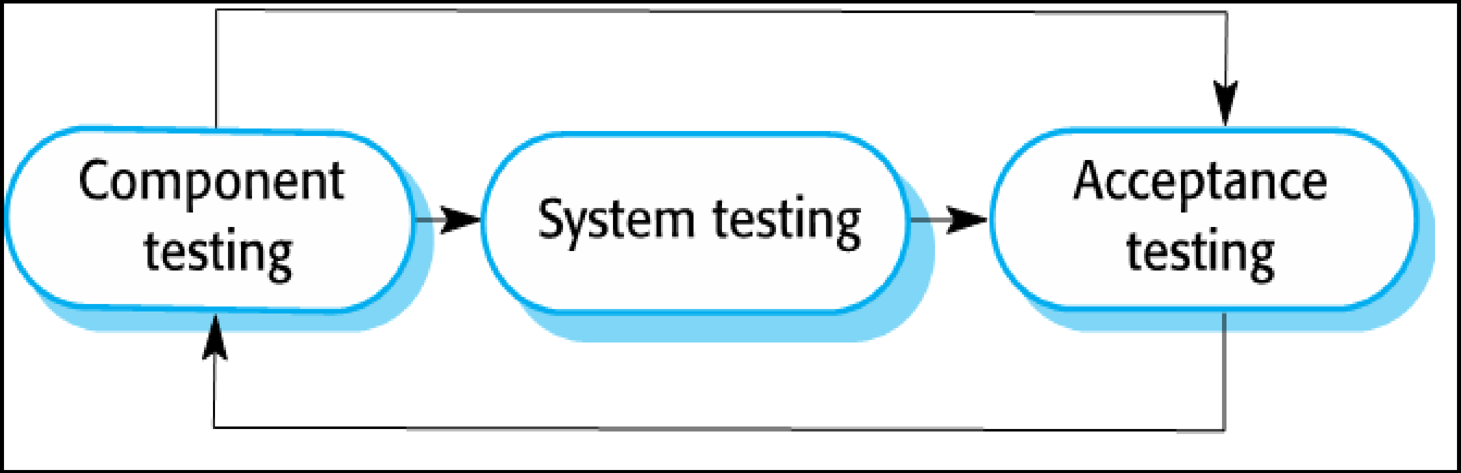
- The software is implemented either by developing a program or programs or by configuring an application system.
- Design and implementation are interleaved activities for most types of software system.
- Programming is an individual activity with no standard process.
- Debugging is the activity of finding program faults and correcting these faults.

Software Validation

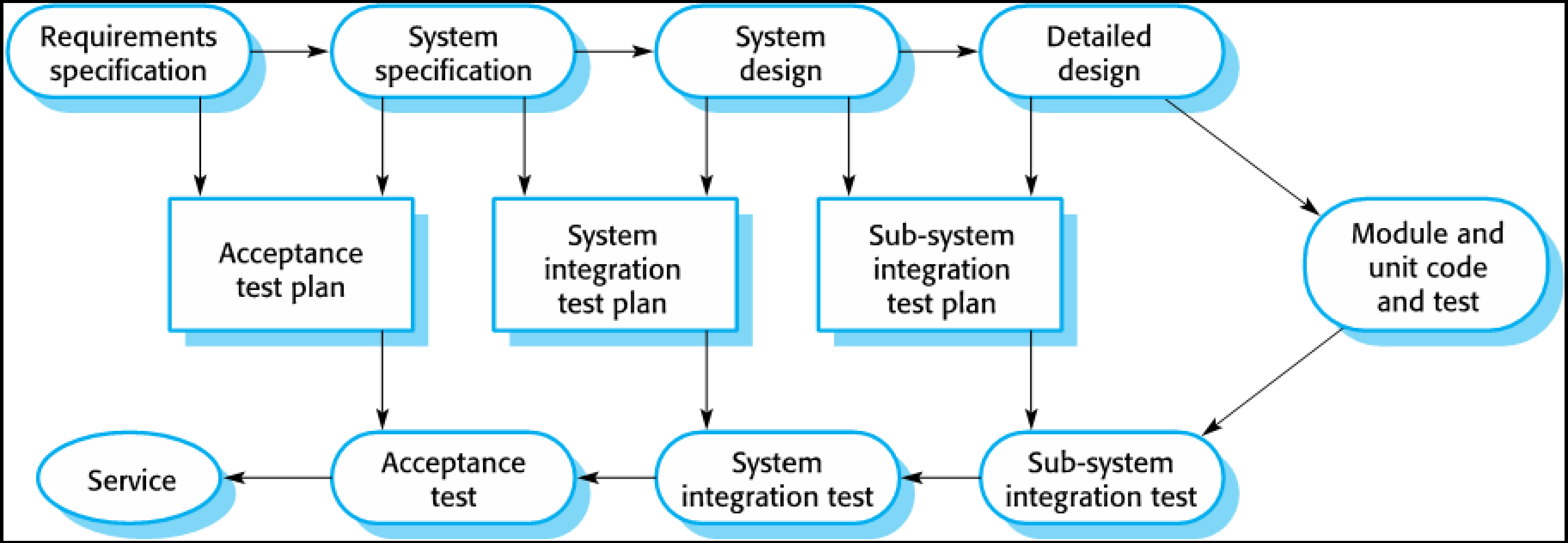
- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.
- Testing is the most commonly used V & V activity.

Testing Stages

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



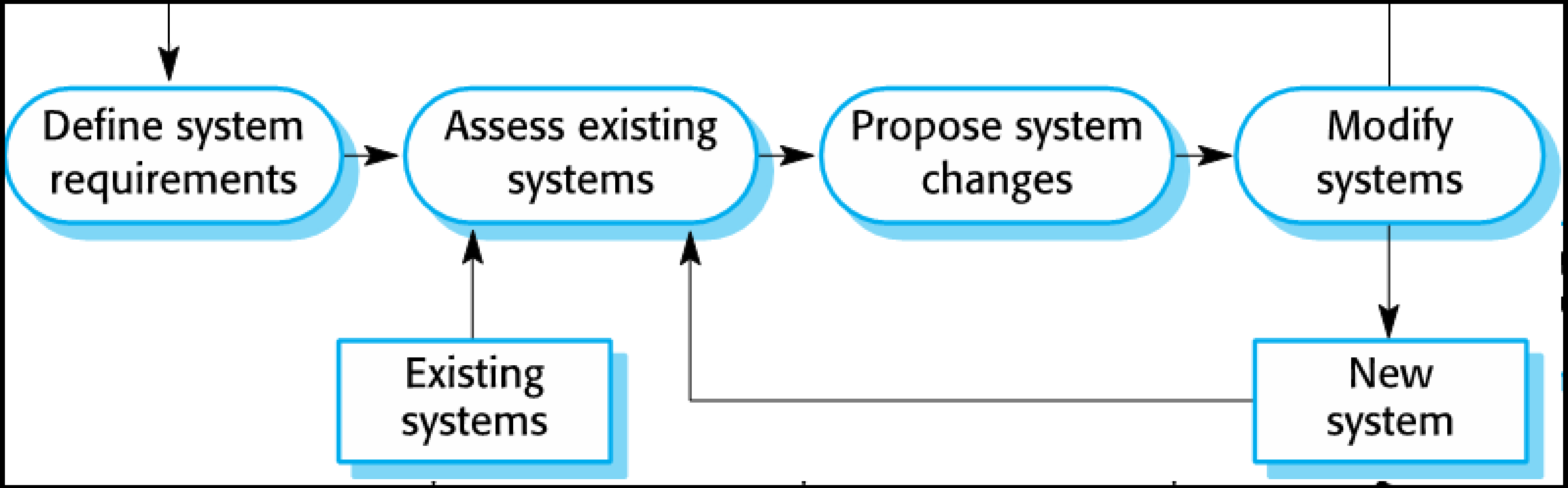
Testing phases in a plan-driven software process (V-model)


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.
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as 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-analysing 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. If this is impossible, then only a single increment (a small part of the system) may have be altered to incorporate the change.

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.

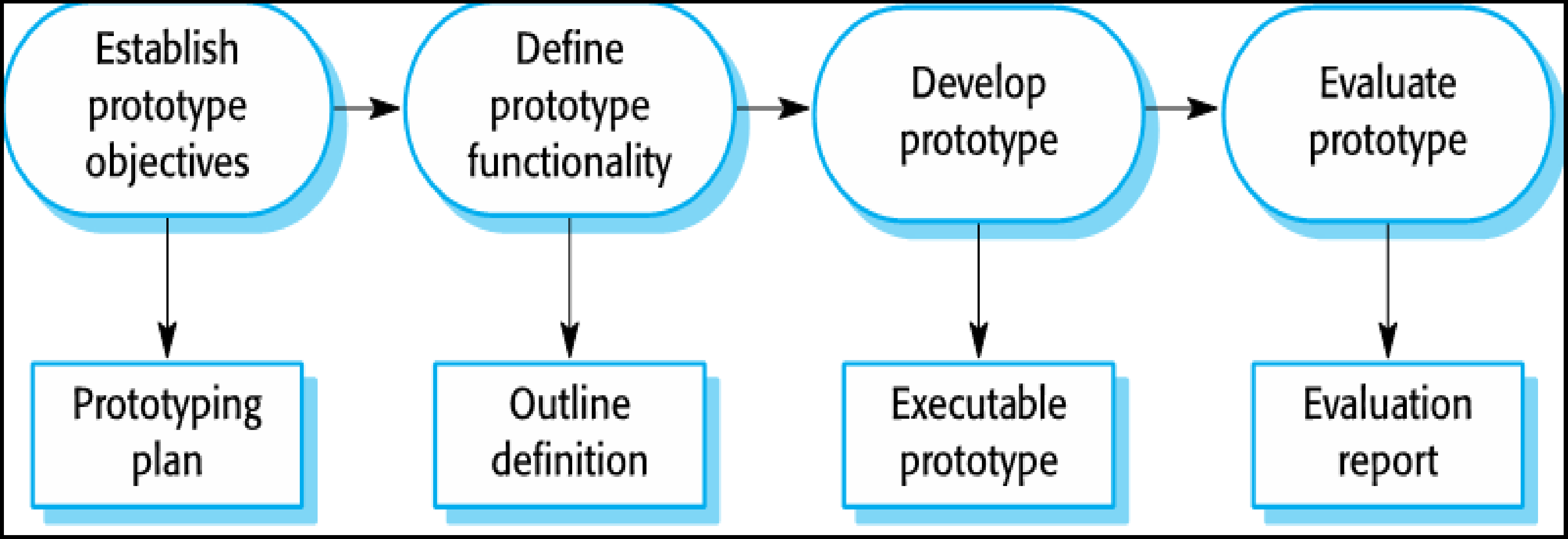
Software Prototyping

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

Benefits of Prototyping

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

The Process of Prototype Development



Prototype Development

- May be based on rapid prototyping languages or tools
- May involve leaving out functionality
  - Prototype should focus on areas of the product that are not well-understood;
  - Error checking and recovery may not be included in the prototype;
  - Focus on functional rather than non-functional requirements such as reliability and security

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 rapid change;
  - The prototype probably will not meet normal organisational quality standards.

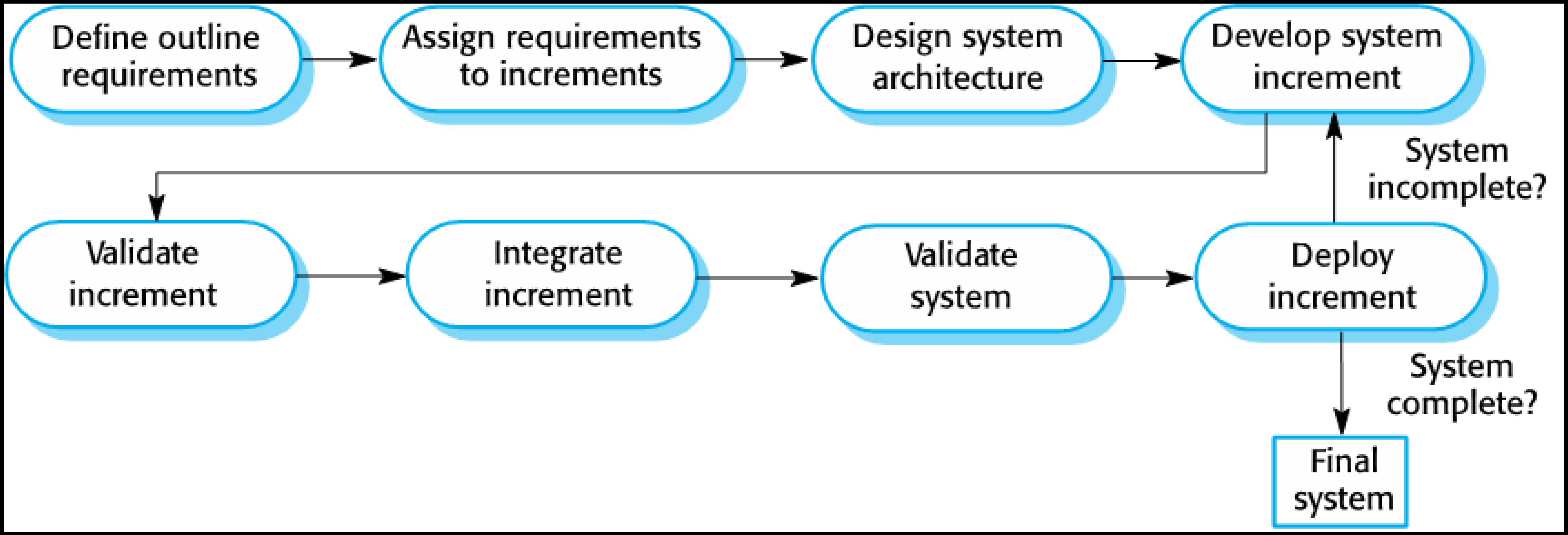
Incremental Delivery

- 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



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

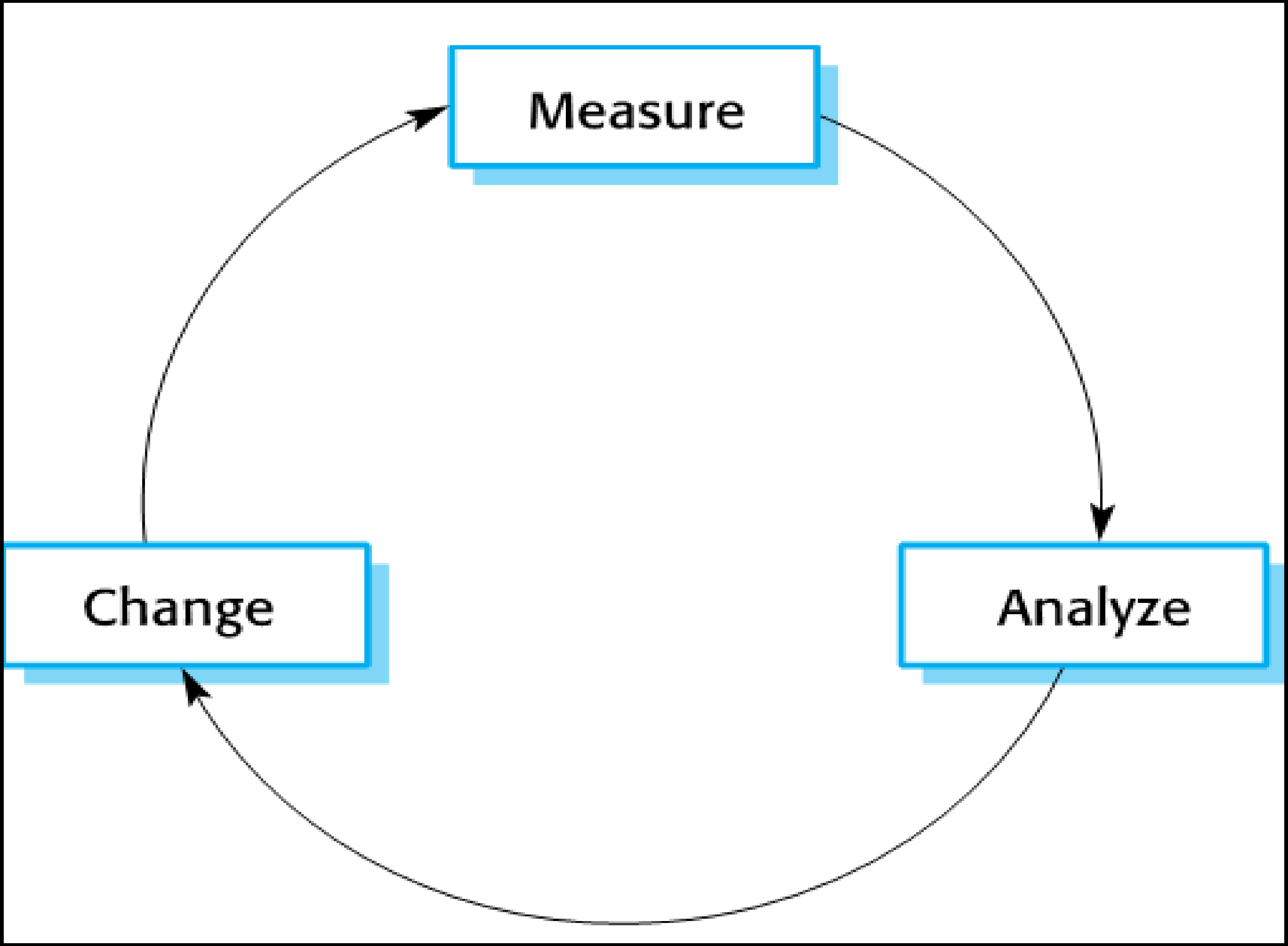
- Most systems require a set of basic facilities that are used by different parts of the system.
  - As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.
- The essence of iterative processes 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

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.

The Process Improvement Cycle



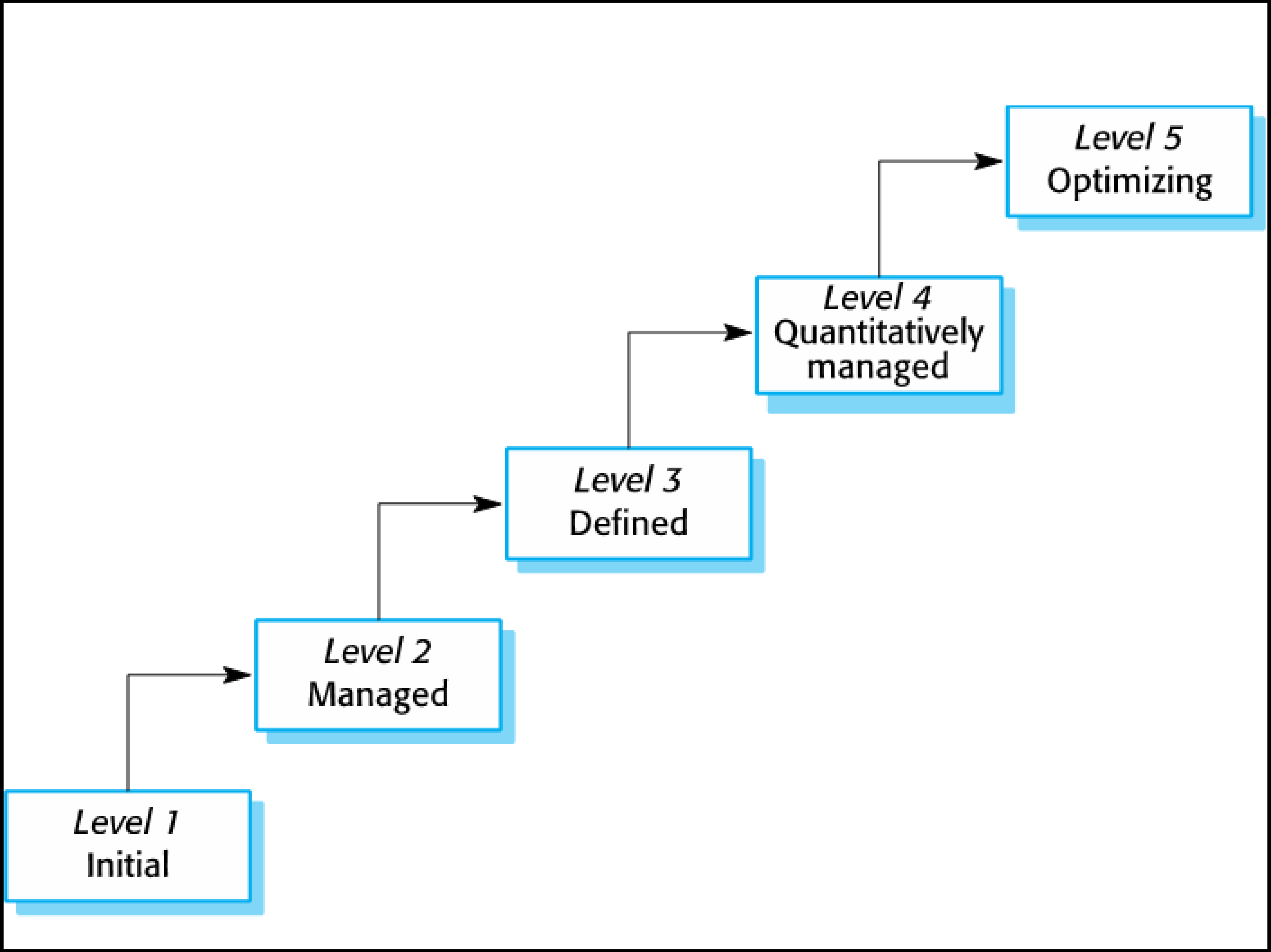
Process Improvement Activities

- *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.

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

That is all