Chapter 2 - Software Processes

MI Qing (Lecturer)

Telephone: 15210503242

Email: miqing@bjut.edu.cn

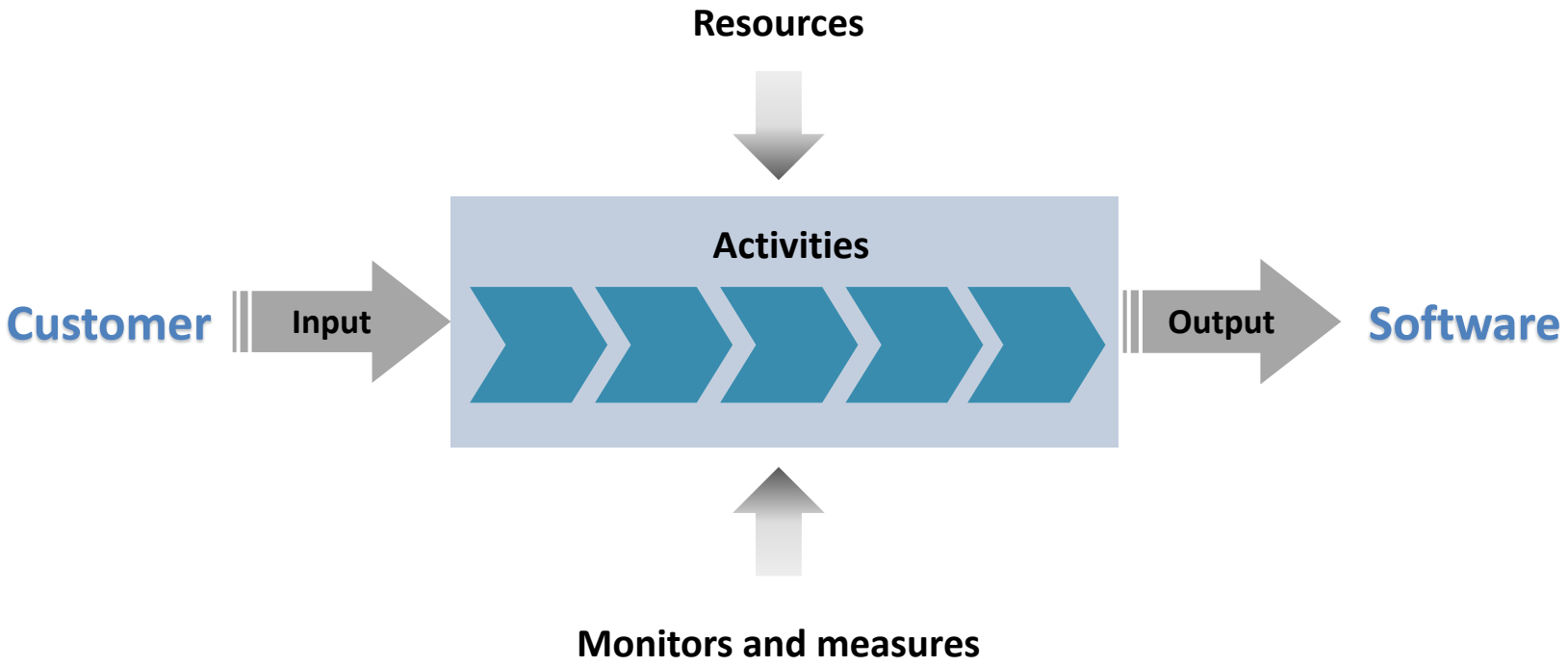
Office: 410, Information Building

Topics Covered

- ✧ Process Activities
- ✧ Software Process Models
- ✧ Process Improvement

Software Process

- A software process is a set of related activities that leads to the production of a software system.



Software Process

- 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
- 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, and the ordering of these activities.

Process Activities

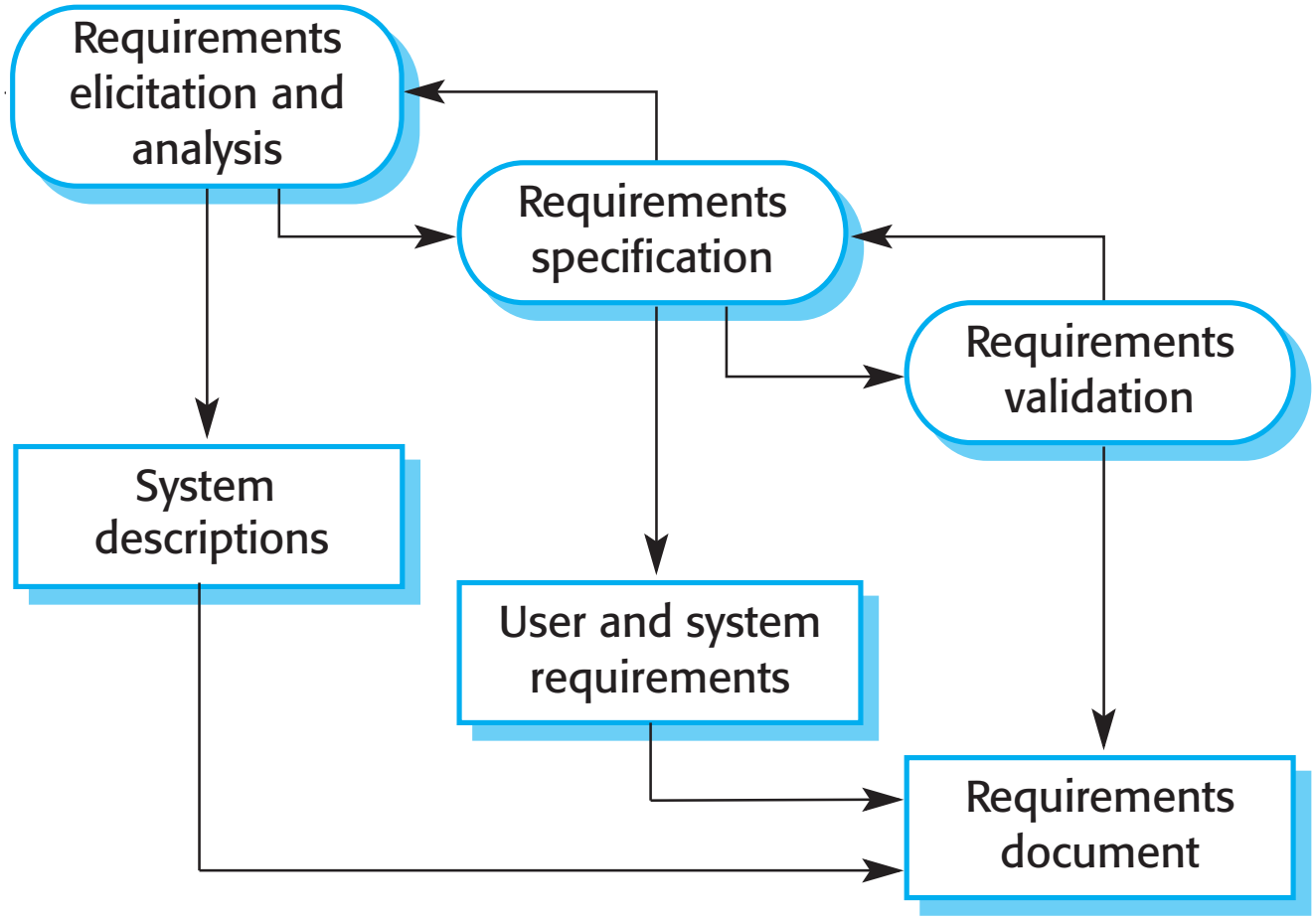
Process Activities

- 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.

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

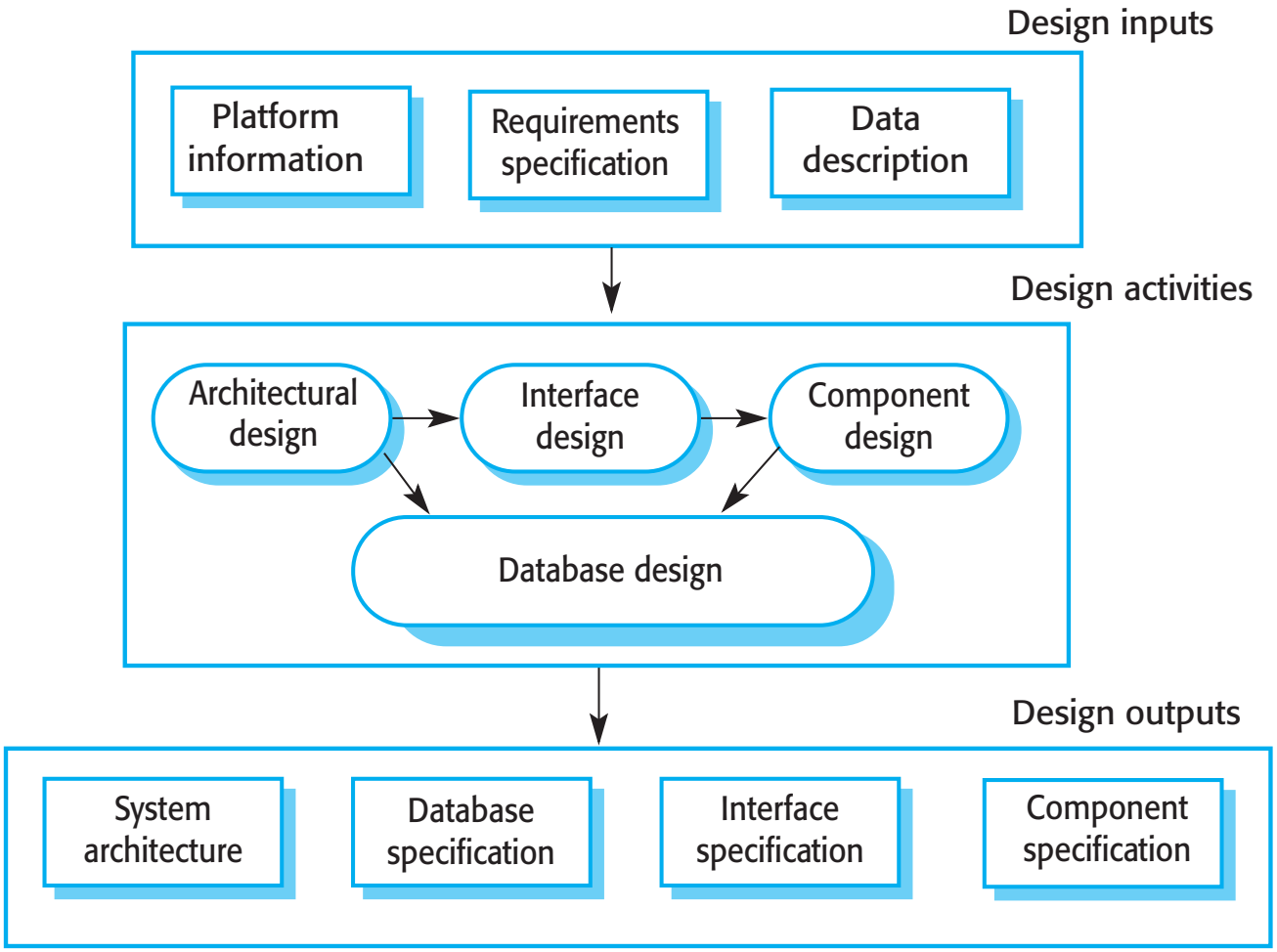
Requirements Engineering Process



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

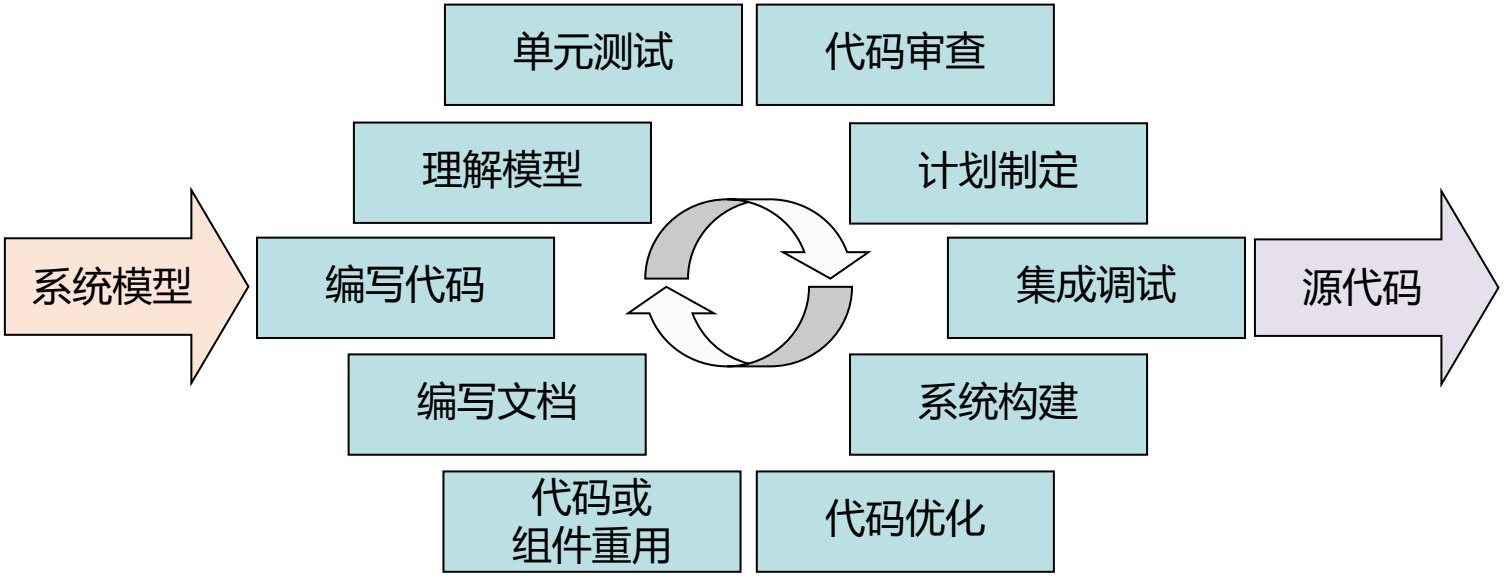


Design Activities

- 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.

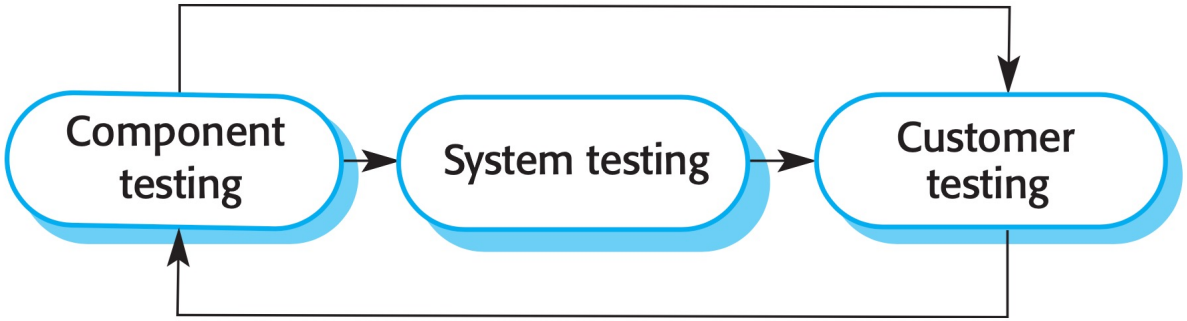
System Implementation

- Design and implementation are interleaved activities for most types of software system.
- Programming is an individual activity with no standard process.



Software Validation

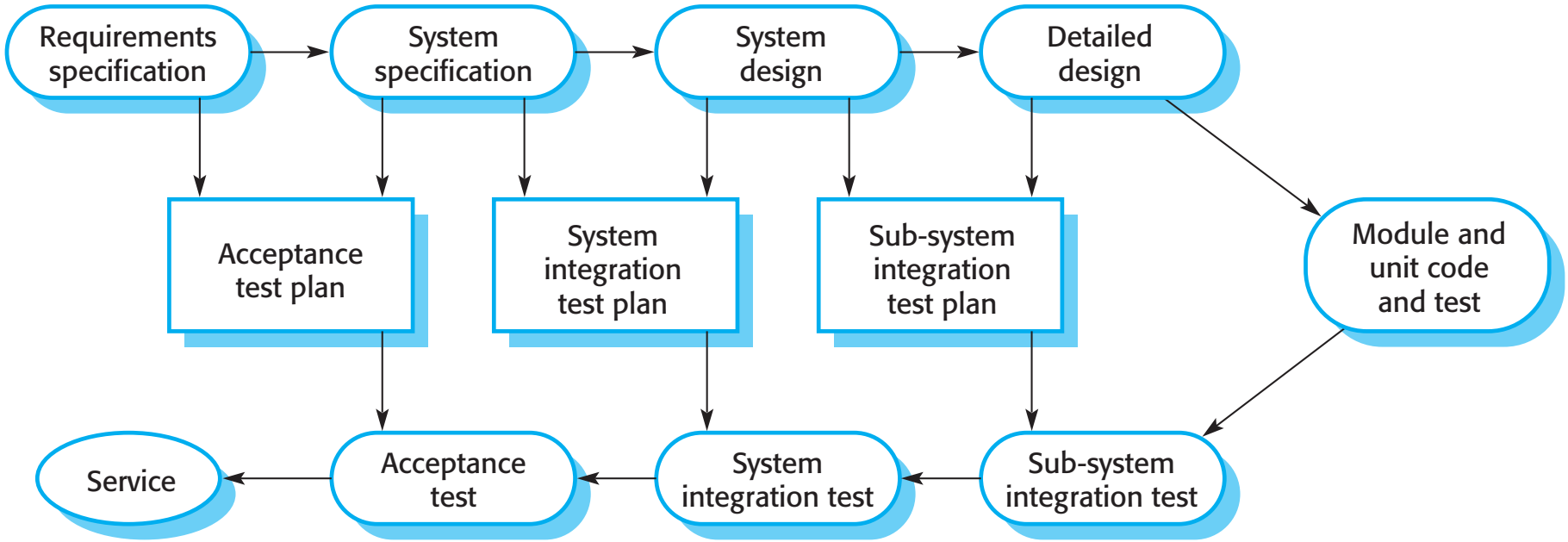
- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.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

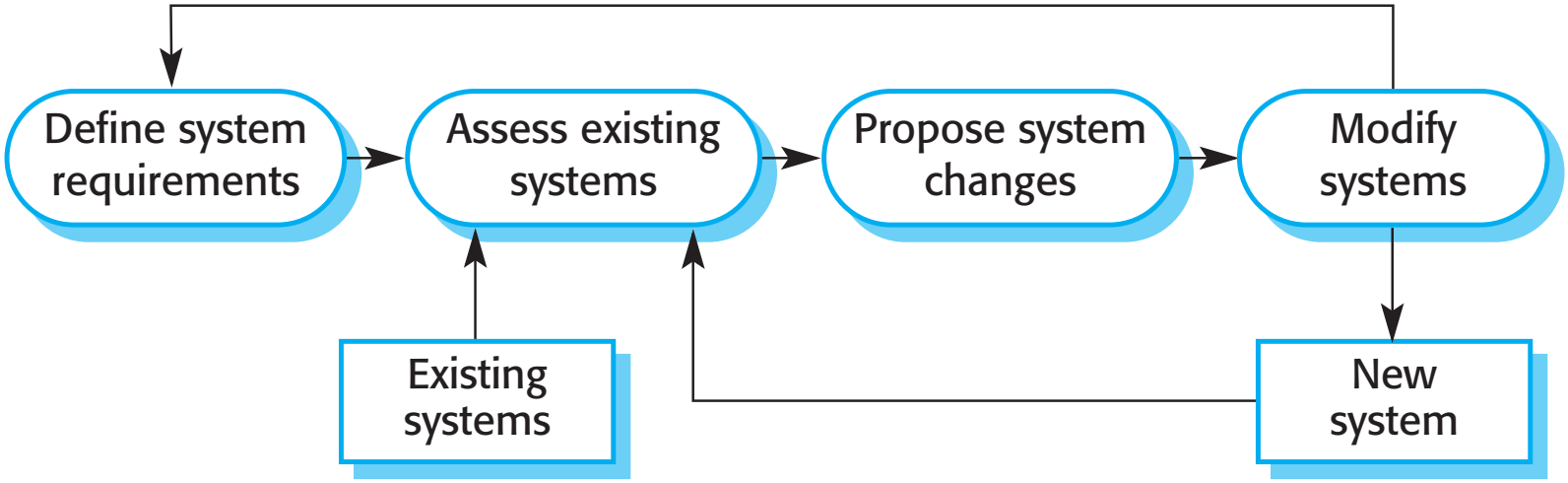
V-model


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.

Software System Evolution



Software Process Models

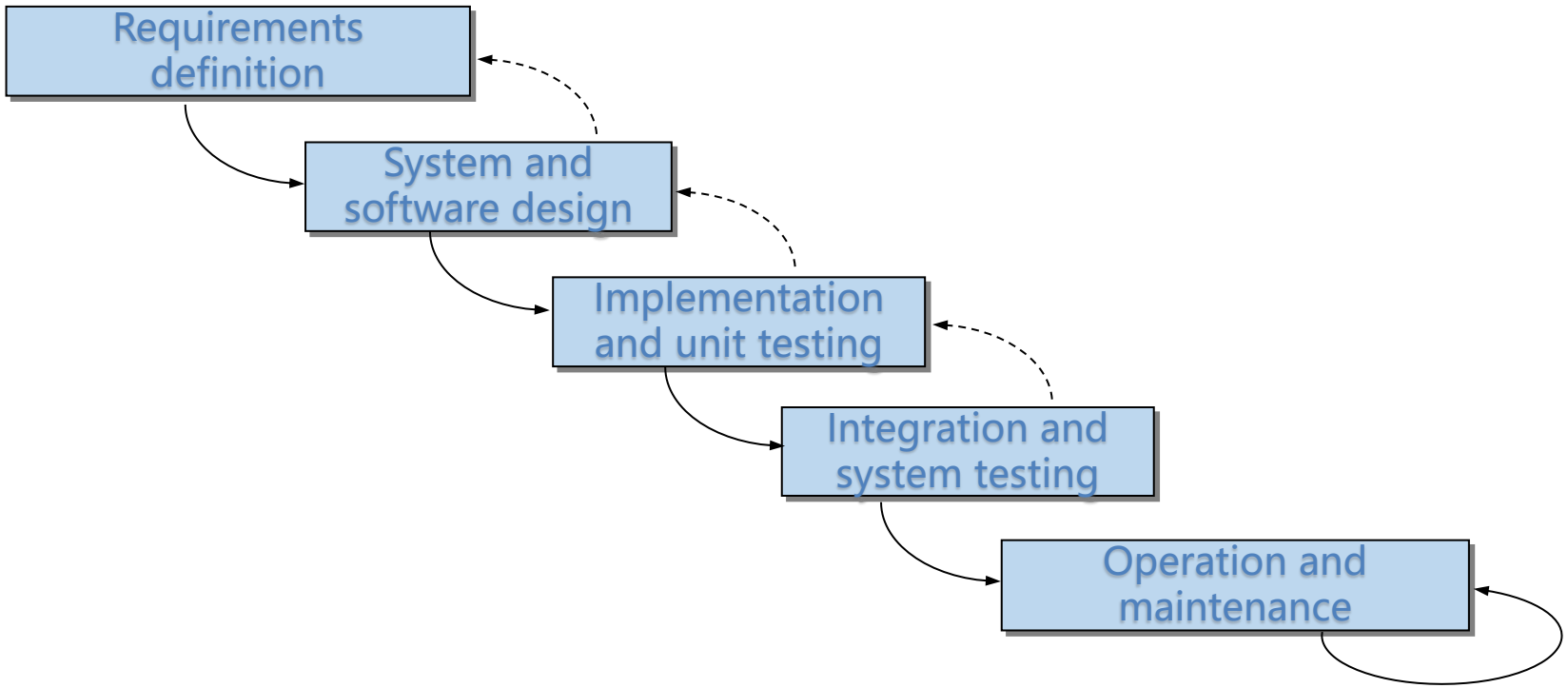
Plan-driven and Agile Processes

- 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

- Waterfall model
  - Plan-driven model. Separate and distinct phases of specification and development.
- Prototype model
- Incremental model
  - Specification, development and validation are interleaved. 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 Model



Waterfall Model Benefits

- ✧ Easy to understand and implement.
- ✧ Reinforces good habits: define-before-design, design-before-code.
- ✧ Identifies deliverables and milestones.
- ✧ Works well on mature products and weak teams.

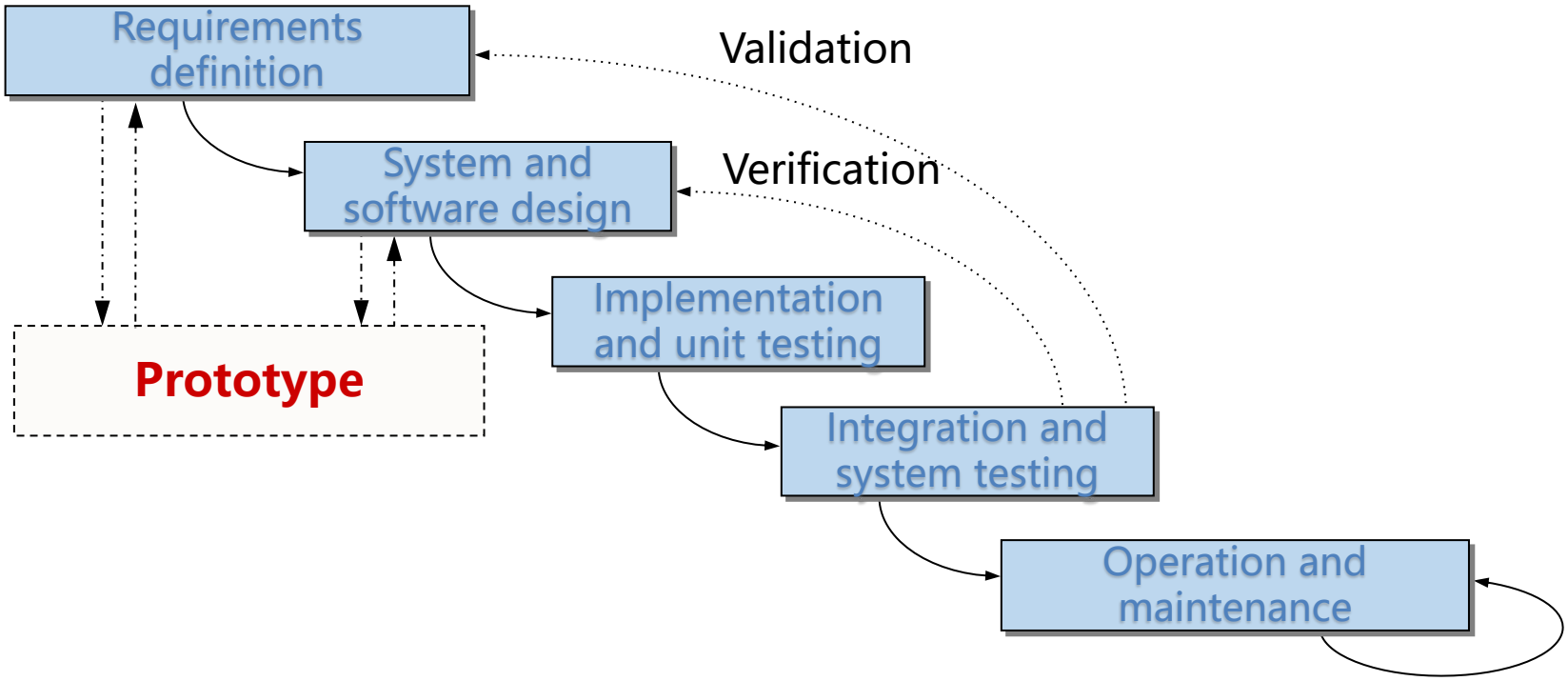
Waterfall Model Problems

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

Coping with Change

- 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.

Prototype Model

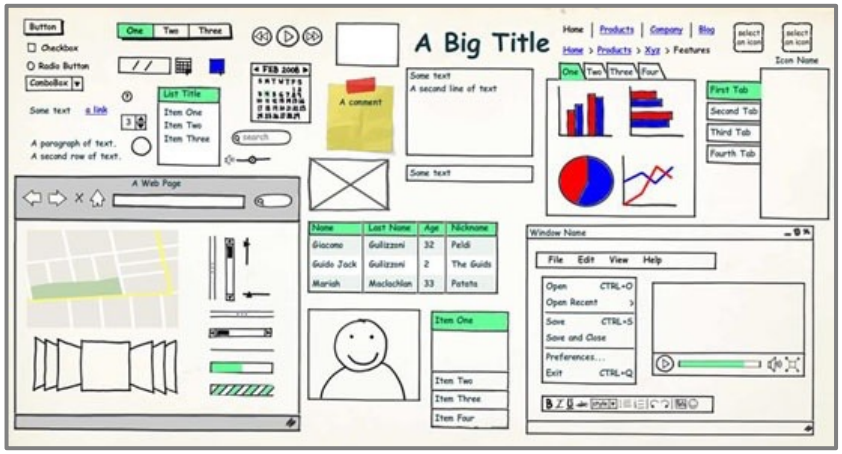


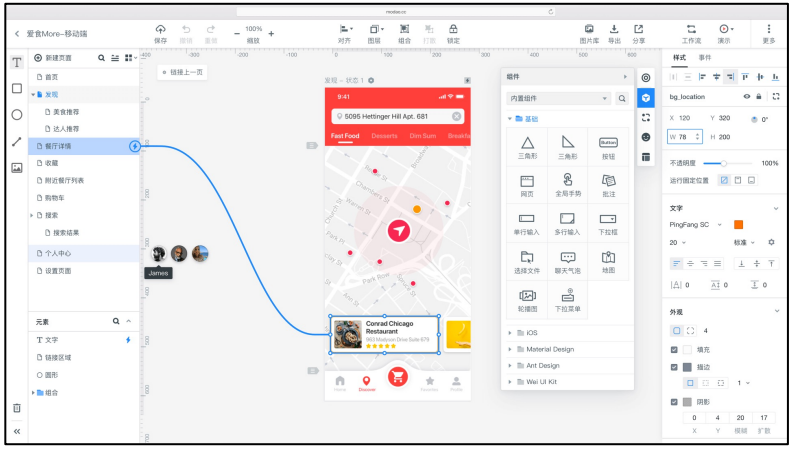
Software Prototyping

- 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
- Benefits of prototyping:
  - Improved system usability
  - A closer match to users' real needs
  - Improved design quality
  - Reduced development effort

Prototype Development

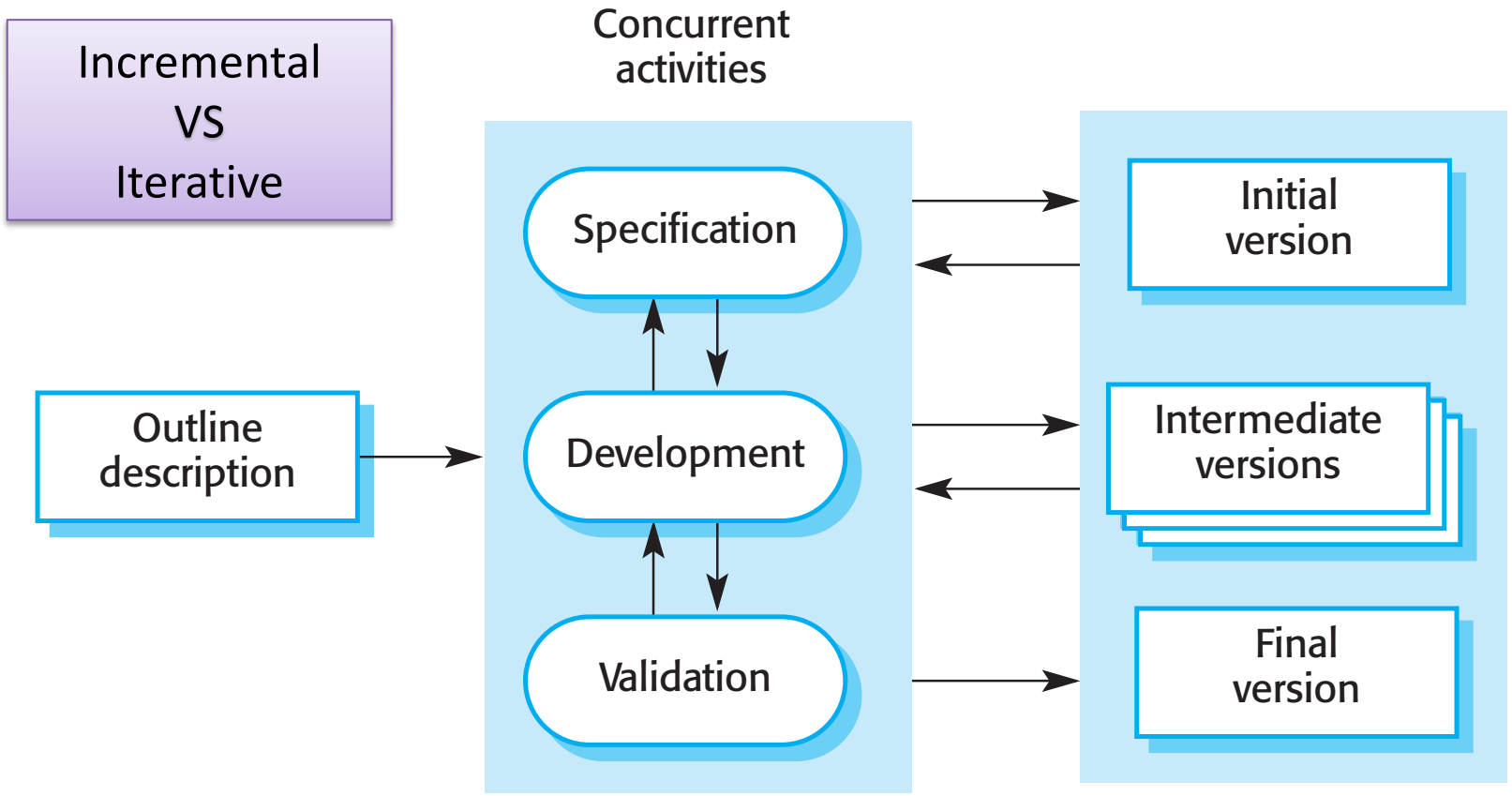
- ✧ Paper Prototyping
- ✧ Tools
  - Balsamiq Mockups、墨刀、Mockplus、Axure RP、InVision





Incremental Model

Incremental
VS
Iterative



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

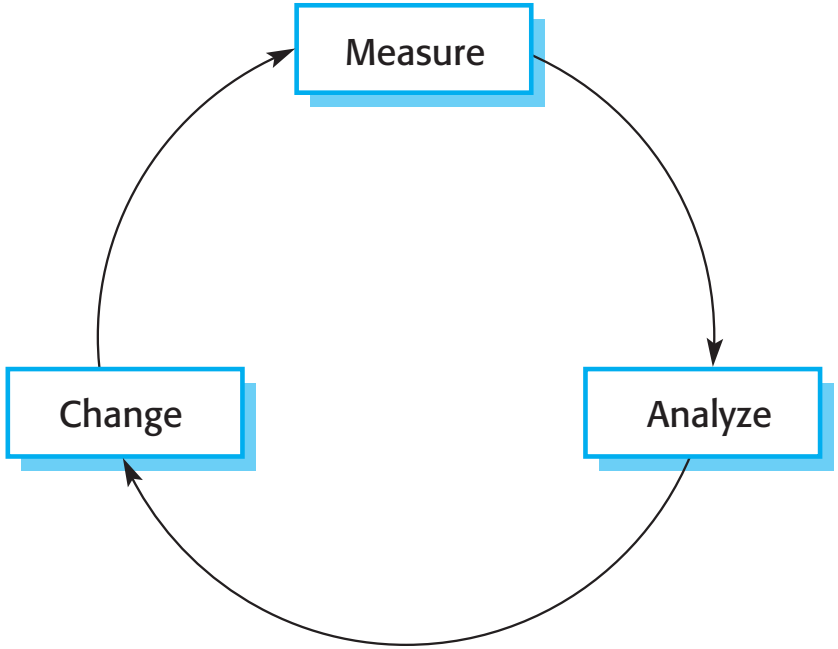
Incremental Model Problems

- 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.
- 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

The Process Improvement Cycle

- Process improvement means understanding existing processes and changing these processes to increase product quality and/or reduce costs and development time.



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

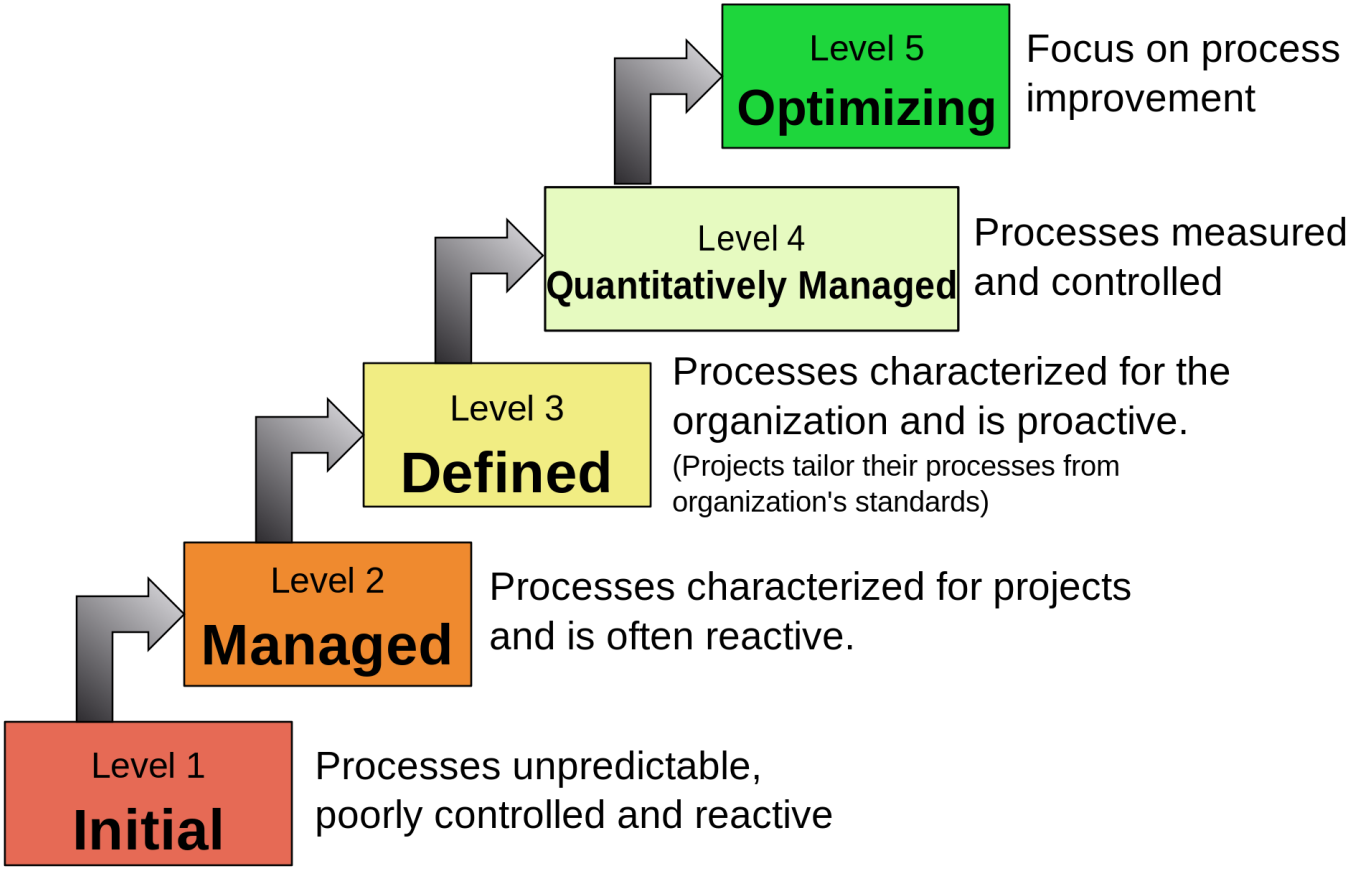
- Process changes are proposed to address some of the identified process weaknesses. These are introduced and the cycle resumes to collect data about the effectiveness of the changes.

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.

Capability Maturity Model (CMM)

Characteristics of the Maturity levels



Summary

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 organization of software processes.
  - Examples of these general models include waterfall model, prototype model, and incremental model.
- Processes should include activities such as prototyping and incremental delivery to cope with change.

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.

Key Points

- The principal approaches to process improvement are agile approaches, geared to reducing process overheads, and maturity-based approaches based on better process management and the use of good software engineering practice.
- The SEI process maturity framework identifies maturity levels that essentially correspond to the use of good software engineering practice.