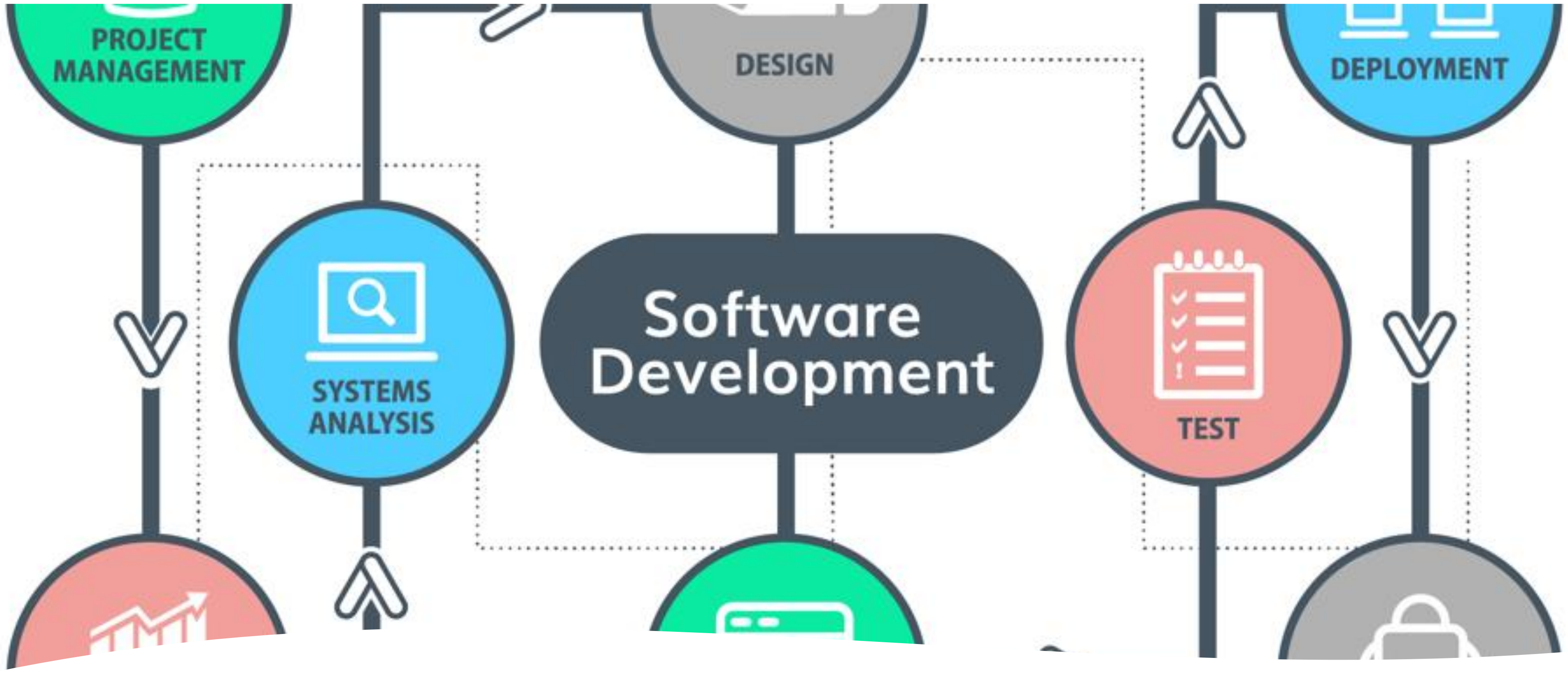
Bachelor of Computer Science

SCS1302:
Introduction to Software Engineering

Software Processes

Prof. K. P. Hewagamage

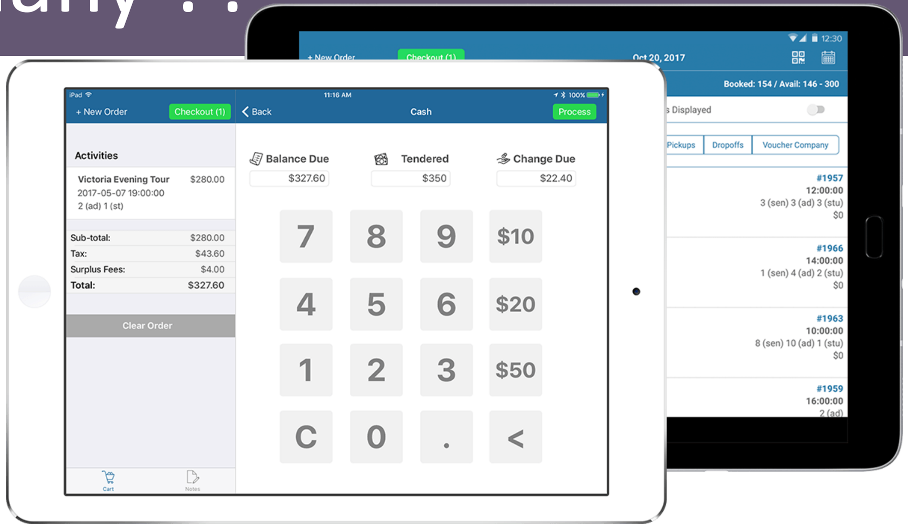


Software Process

- Is there one Process , One Road Map, One predefine template that will take through all the necessary steps from start to finish to make a piece of software?

NO there are many

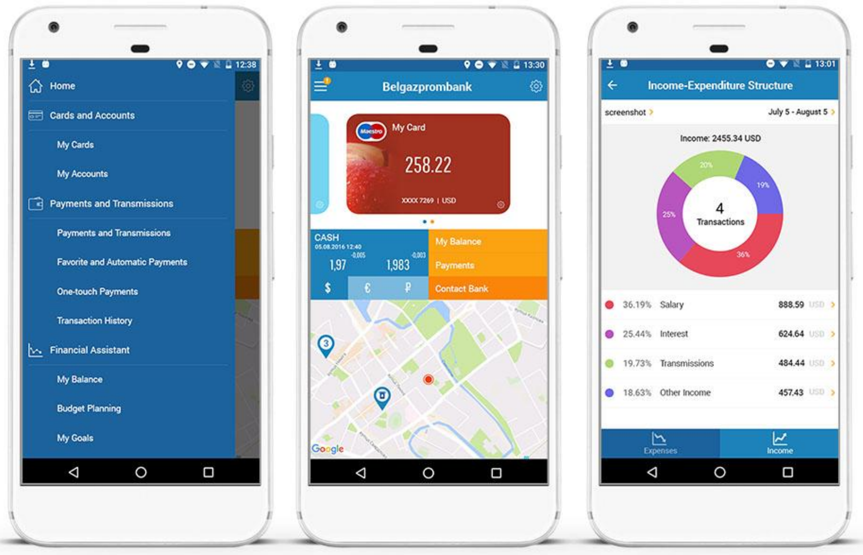
Why so many ??



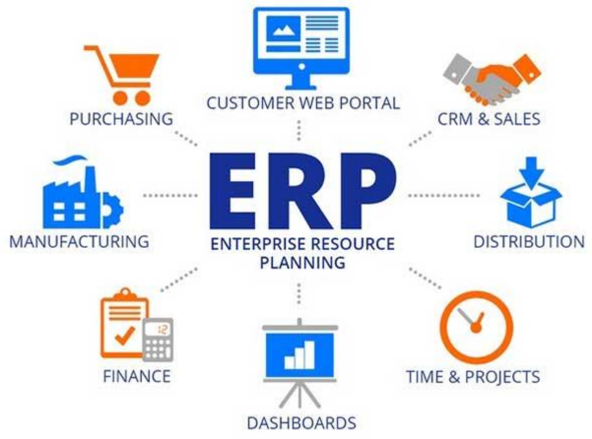
Victoria Evening Tour $280.00
2017-05-07 19:00:00
2 (ad) 1 (st)

Sub-total:	$280.00
Tax:	$43.60
Surplus Fees:	$4.00
Total:	$327.60

Balance Due	Tendered	Change Due
$327.60	$350	$22.40



36.19%	Salary	888.59 USD
25.44%	Interest	624.64 USD
19.73%	Transmissions	484.44 USD
18.63%	Other Income	457.43 USD



Software Development Processes

- Is a set of related activities that lead to the production of a software product.

  Activities may involve:

  - Development of software from scratch (using C++,Java)
  - Development by extending and modify existing software
  - Configuring and integrating off-the-shelf software or components
- Many different Software Processes exist

Software Development Processes

- All Processes must include four main Phases
    - Software Specification
    - Software Design and Implementation
    - Software Validation
    - Software evolution
- Phases will have activities
    - Eg. Establishing a Database
- Activities will have tasks
    - Eg. Writing a piece of source code

Learning Outcomes

- Appreciate the need for a defined software process
- Be able to describe in detail the main software process models
- Be able to compare software process models
- Identify the most appropriate software process model for a given problem
- Identify how CASE tools can be used to support software process activities

Road Map

1. Software process models
2. Process activities
3. Coping with change
4. Process Improvement

Road Map

1. **Software process models**
2. Process activities
3. Coping with change
4. Process Improvement

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.

The software process

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

- **Why do we need a process model?**

Why do you need software process model

when a team writes down a description of its development process it forms a common understanding of the activities, resources and constraints involved in software development

creating a process model helps the team find inconsistencies, redundancies and commissions in the process, as these problems are noted and corrected the process becomes more effective

Why do you need software process models?

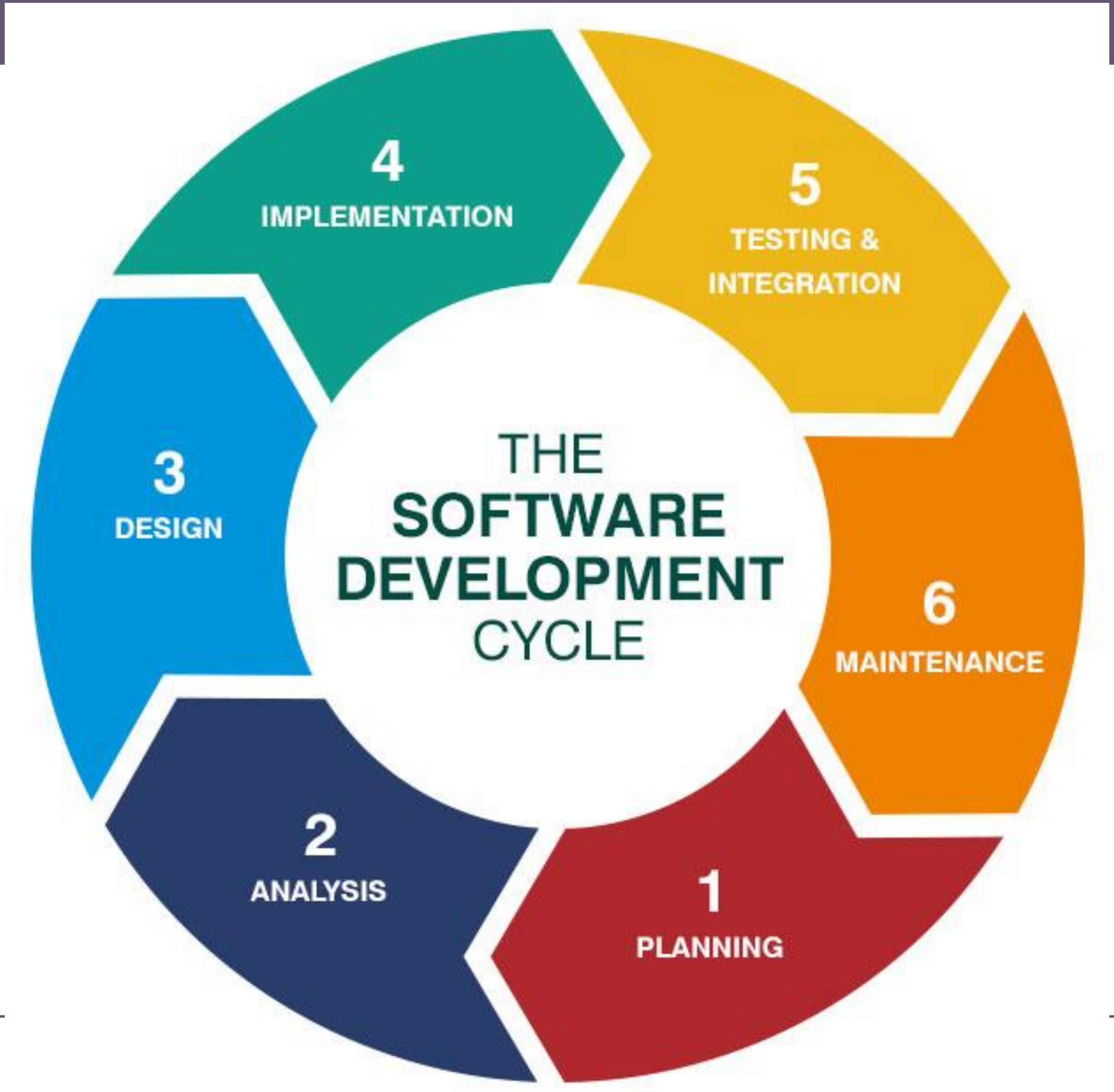
- the model reflects the goals of development and shows explicitly how the product characteristics are to be achieved

- each development is different and a process has to be tailored for different situations, the model helps people to understand these differences

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 - outcomes of a process activity;
  - Roles - responsibilities of the people (Software Engineer, QA Engineer, Project Manager)
  - Pre- and post-conditions, which are statements that are true before and after a product produced

System Development Life Cycle (SDLC)



Plan-driven and agile processes

- Plan-driven processes- all of the process activities are planned in advance and progress is measured against this plan.
- 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.

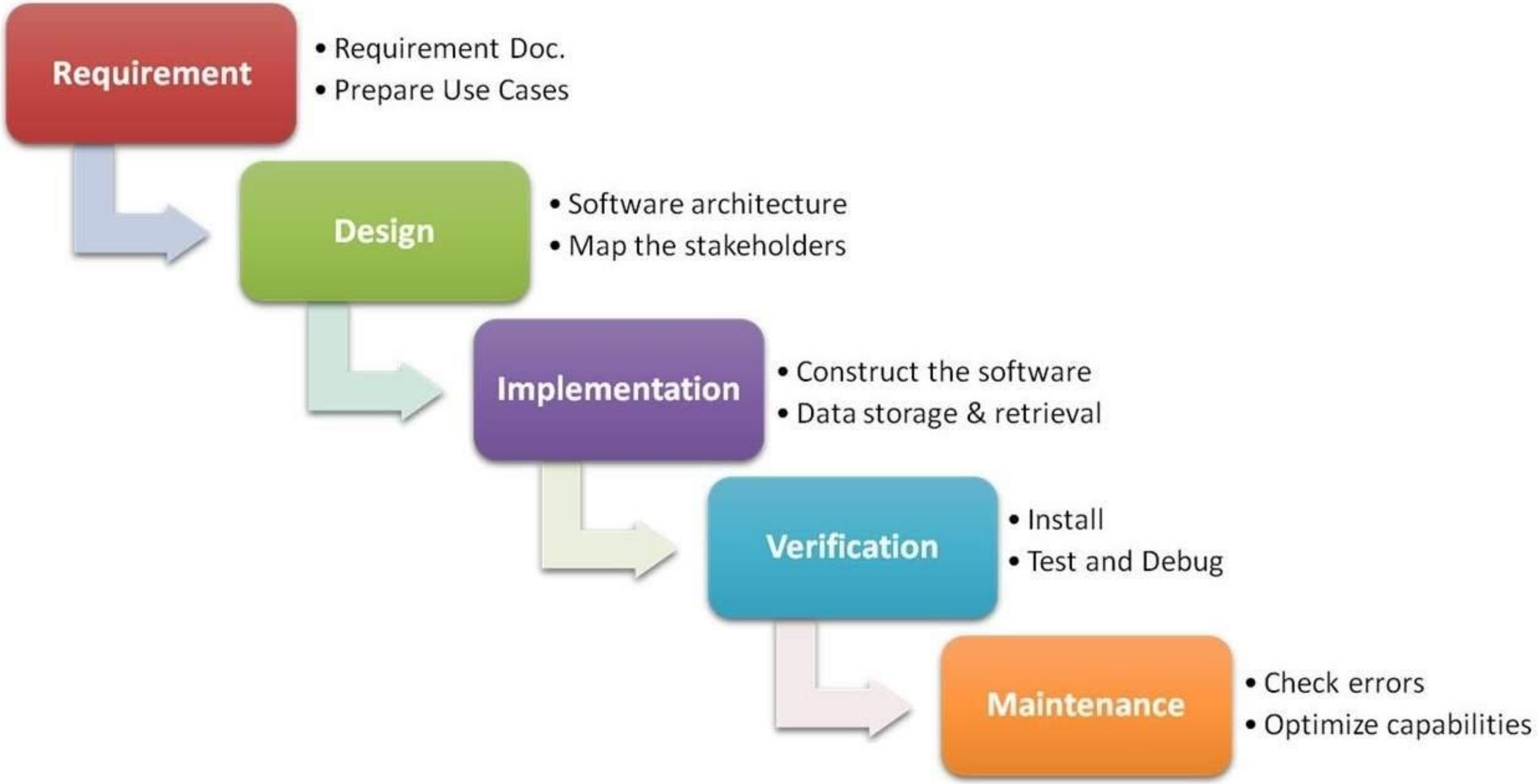
Software process models

- 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.
- Iterative Development
- Reuse-oriented software engineering or (*Integration and configuration)*
  - The system is assembled from existing 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.

Software process models

- Waterfall model
- Prototyping models
- Evolutionary models
- The spiral model
- Formal development
- Incremental development
- Rapid Application Development
- Unified Process
- Agile Process
- Extreme Programming (XP)

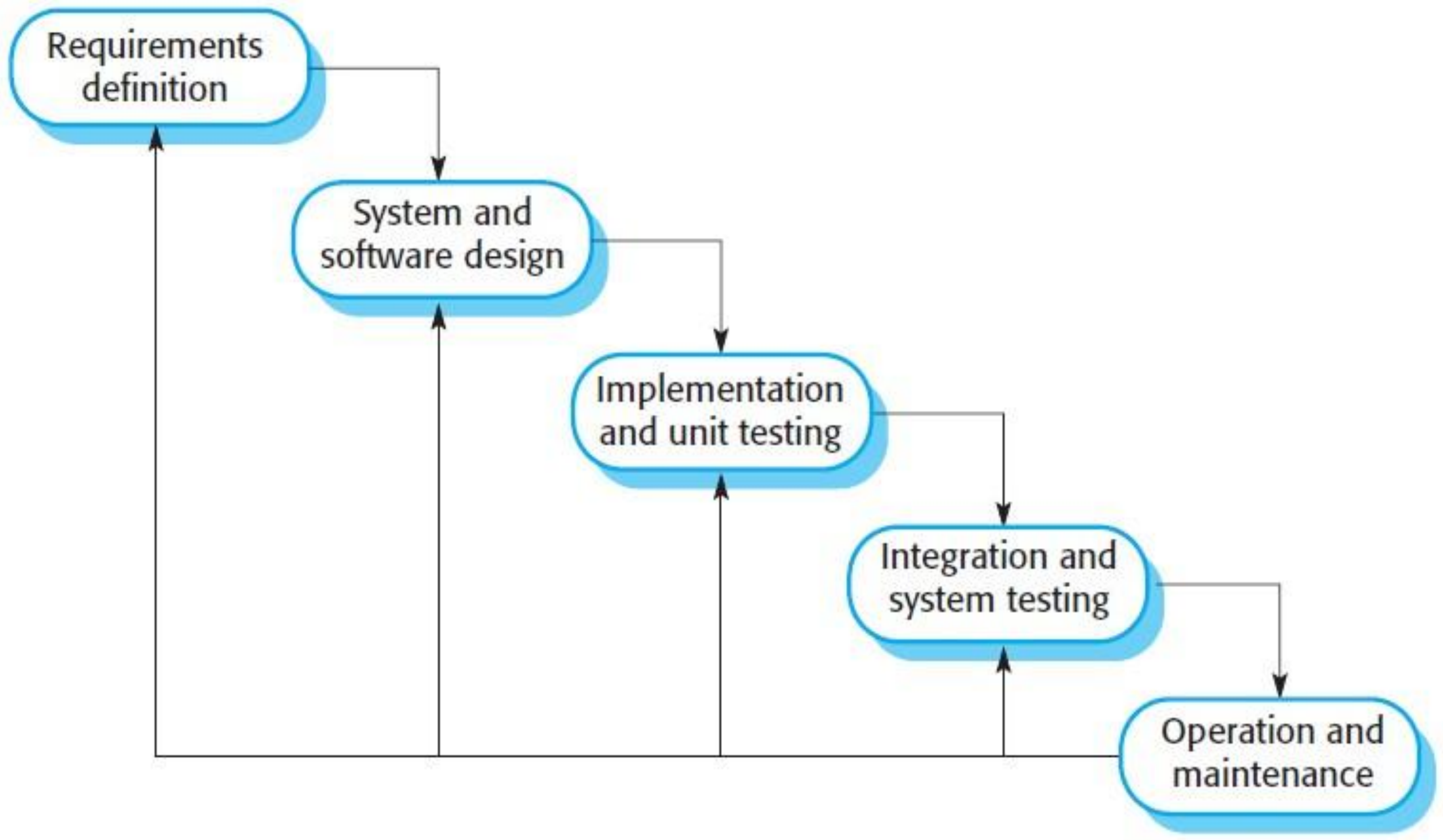
Waterfall Model



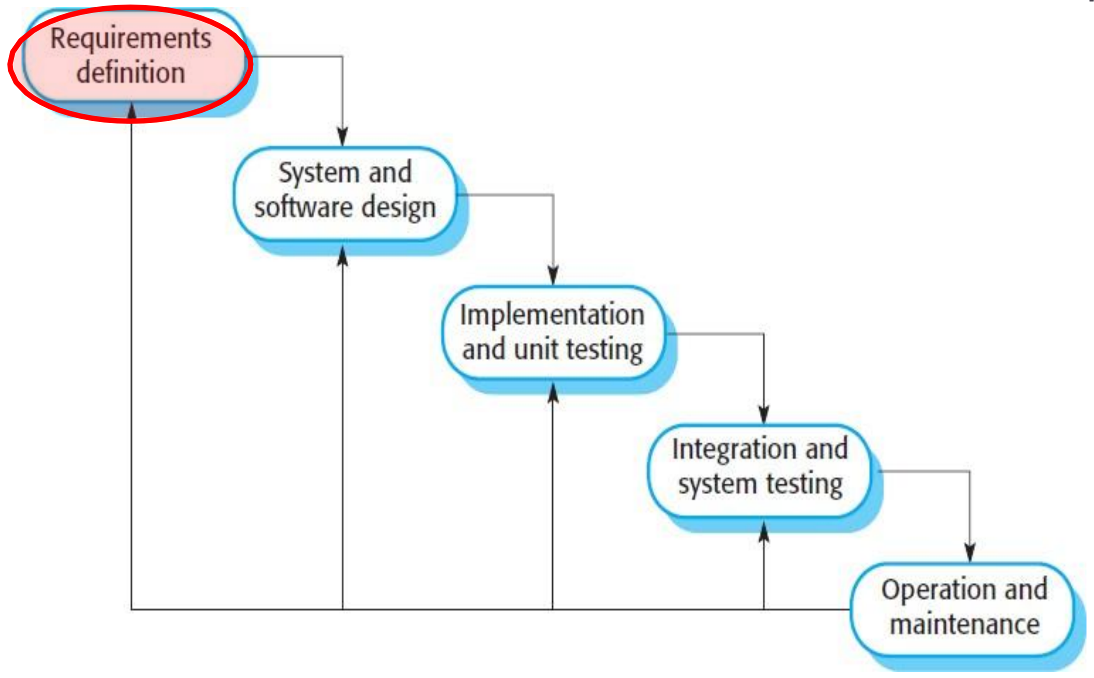
Waterfall Model

- The first published model of the software development process
- The waterfall model is an example of a plan-driven process.
- In principle at least, you plan and schedule all of the process activities before starting software development.

Waterfall Model


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

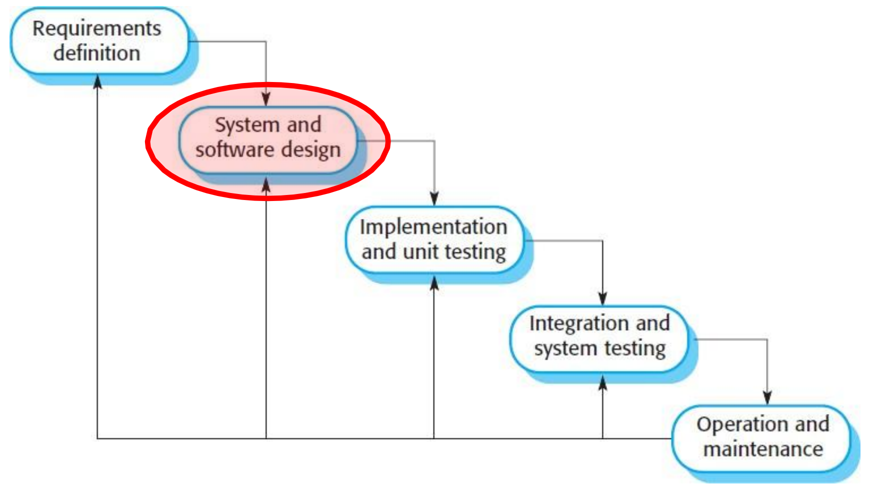
Waterfall Model


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

✧Requirements analysis and definition

- The system's services, constraints, and goals are established by consultation with system users.
- Need to define those in detail and serve as a system specification.

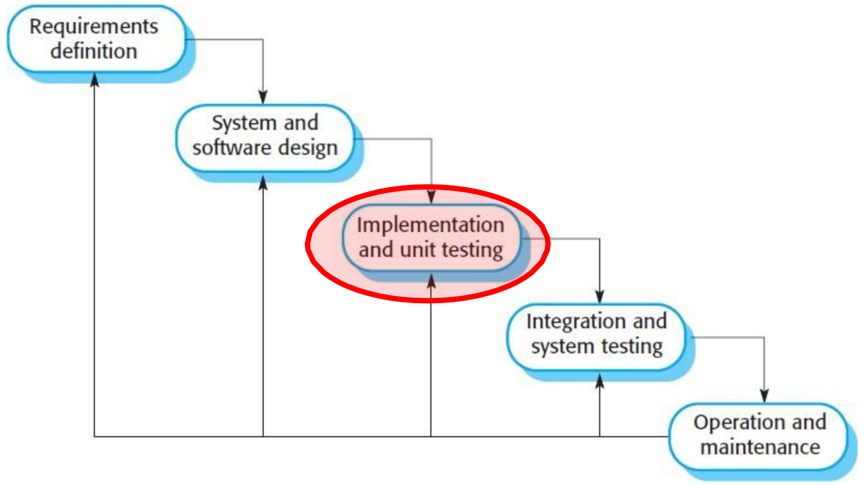
Waterfall Model


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

✧ System and software design

- Allocates the requirements to either hardware or software systems
- It establishes an overall system architecture.
- Software design involves identifying and describing the fundamental software system abstractions and their relationships.

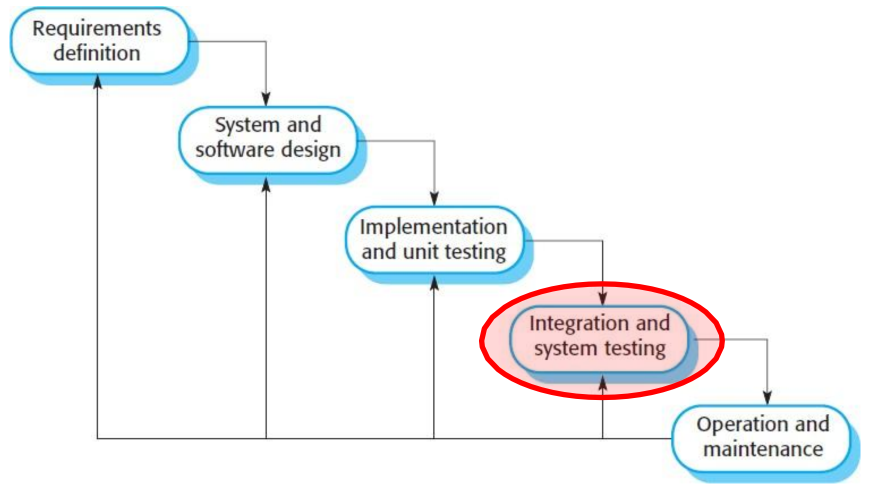
Waterfall Model


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

✧Implementation and unit testing

- During this stage, the software design is realized as a set of programs or program units.
- Unit testing involves verifying that each unit meets its specification.

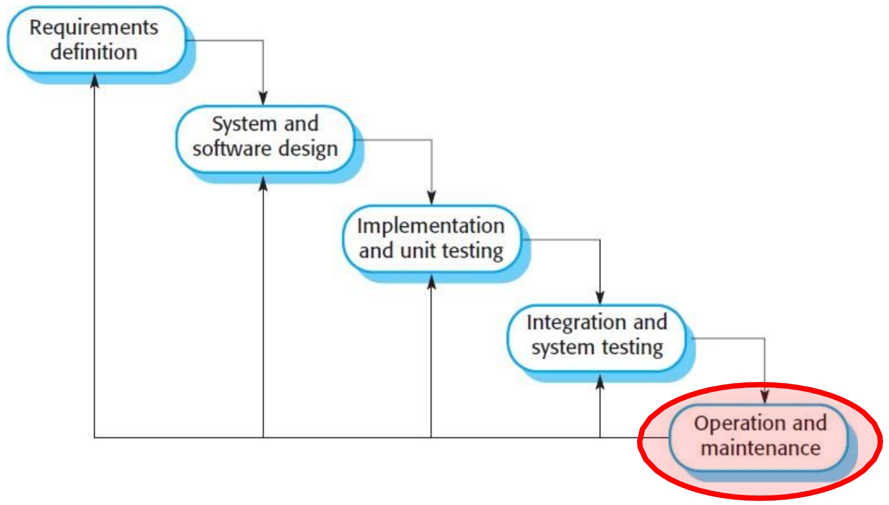
Waterfall Model


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

✧ Integration and system testing

- The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met.
- After testing, the software system is delivered to the customer.

Waterfall Model


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

✧ Operation and maintenance

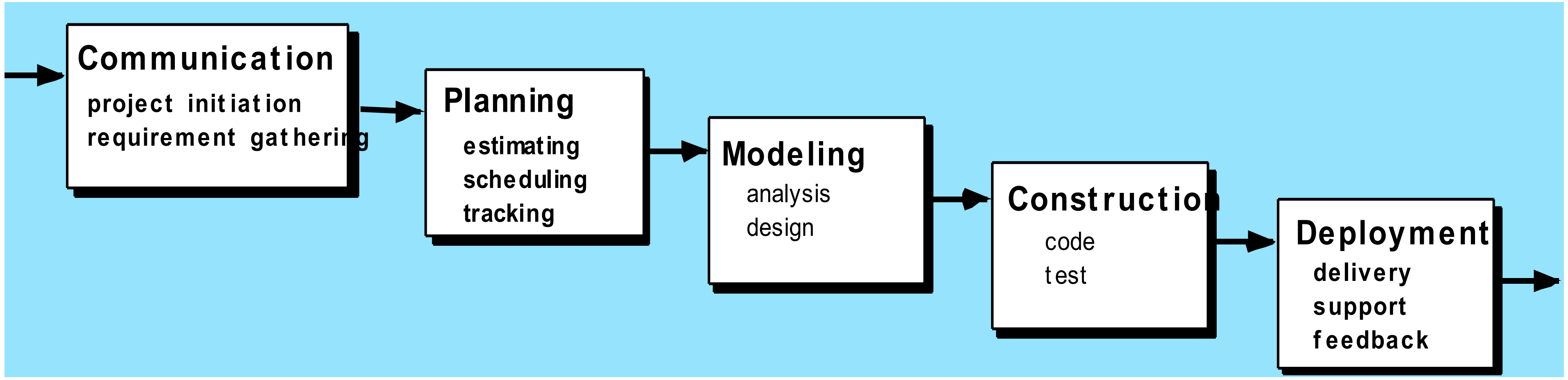
- Normally, this is the longest life- cycle phase.
- The system is installed and put into practical use.
- Maintenance involves
  - correcting errors that were not discovered in earlier stages of the life cycle
  - improving the implementation of system units
  - enhancing the system's services as new requirements are discovered.

Waterfall Model

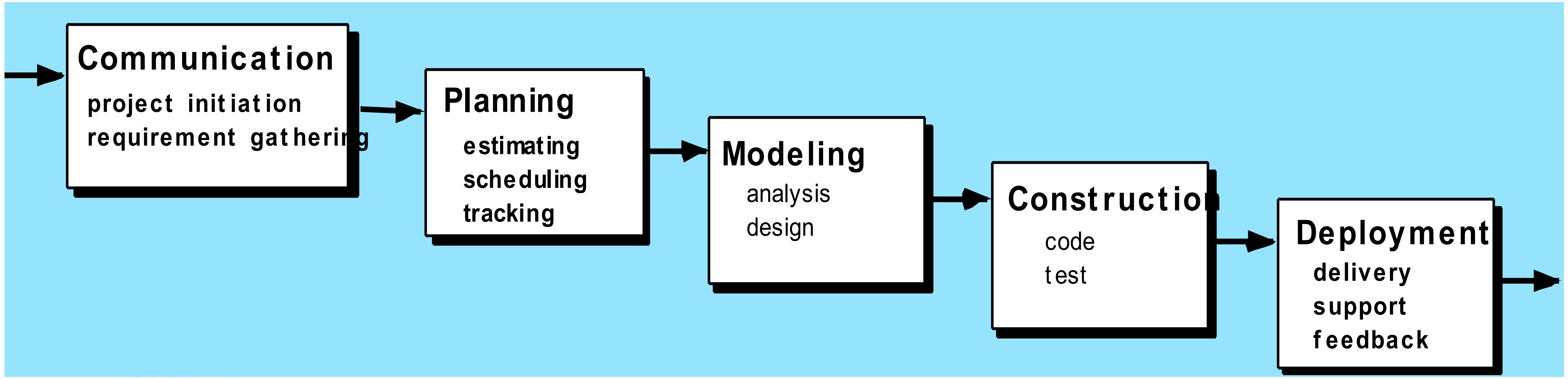
- Waterfall model can be used when
  - Requirements are not changing frequently
  - Application is not complicated and big
  - Project is short
  - Requirement is clear
  - Environment is stable
  - Technology and tools used are not dynamic and is stable
  - Resources are available and trained

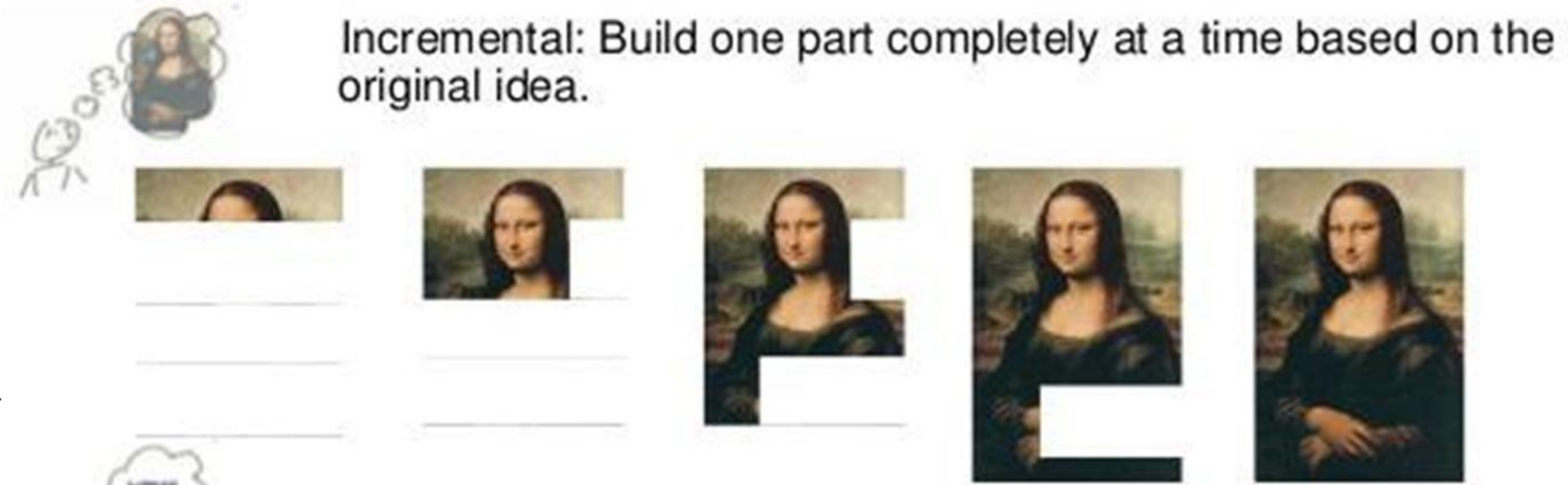
Advantages	Dis-Advantages
• Before the next phase of development, each phase must be completed	• Error can be fixed only during the phase
• Suited for smaller projects where requirements are well defined	• It is not desirable for complex project where requirement changes frequently
• They should perform quality assurance test (Verification and Validation) before completing each stage	• Testing period comes quite late in the developmental process
• Elaborate documentation is done at every phase of the software's development cycle	• Documentation occupies a lot of time of developers and testers
• Project is completely dependent on project team with minimum client intervention	• Clients valuable feedback cannot be included with ongoing development phase

Development/Project Phases based on Waterfall Model



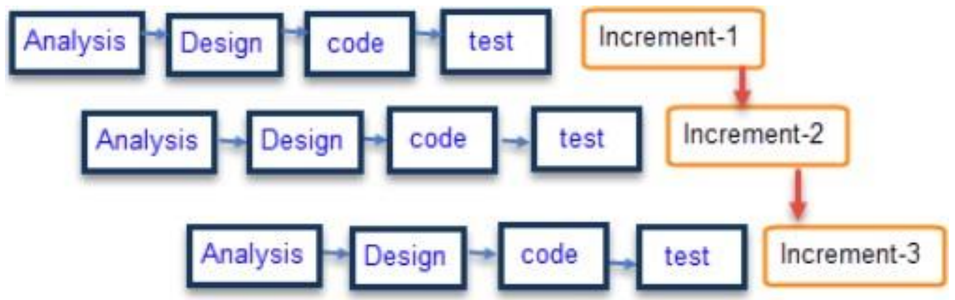
Development/Project Phases based on Waterfall Model



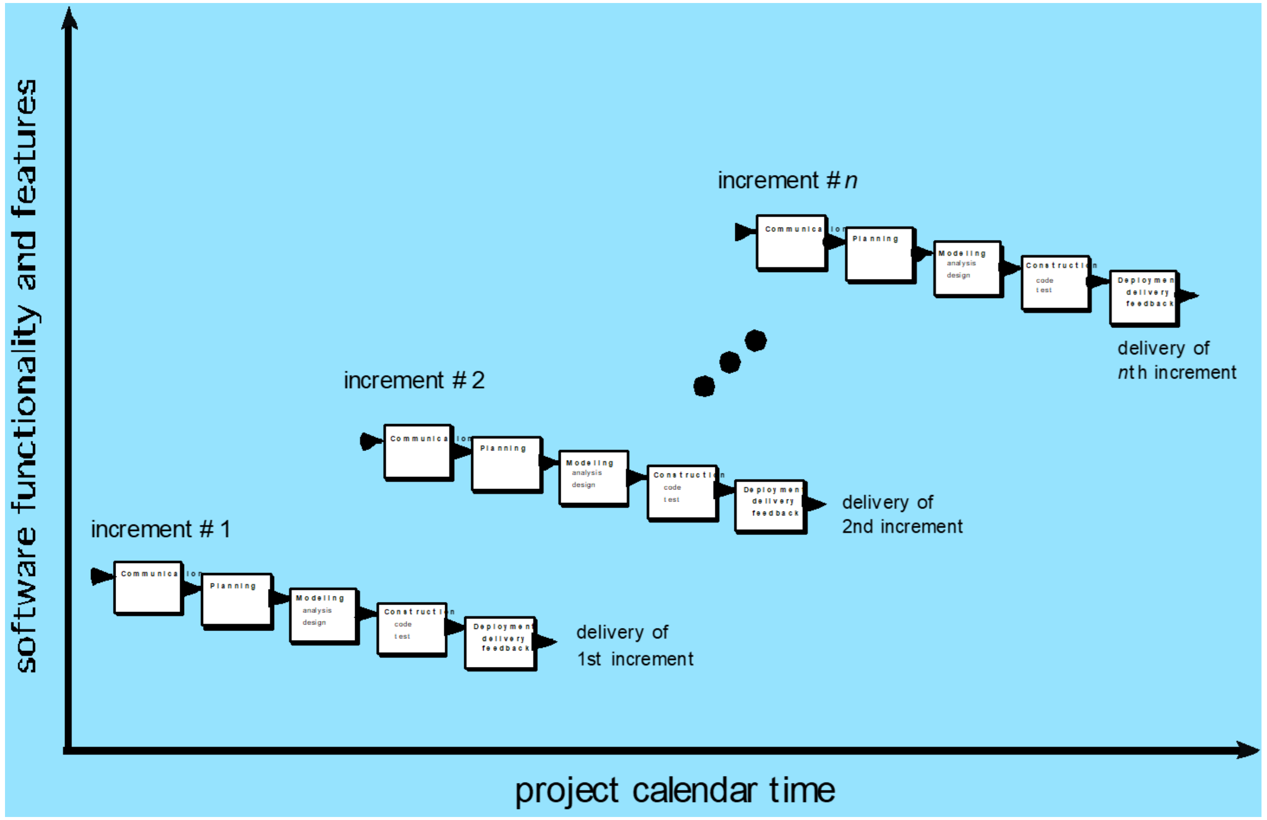

Incremental: Build one part completely at a time based on the original idea.

Incremental Approach

Stages of development



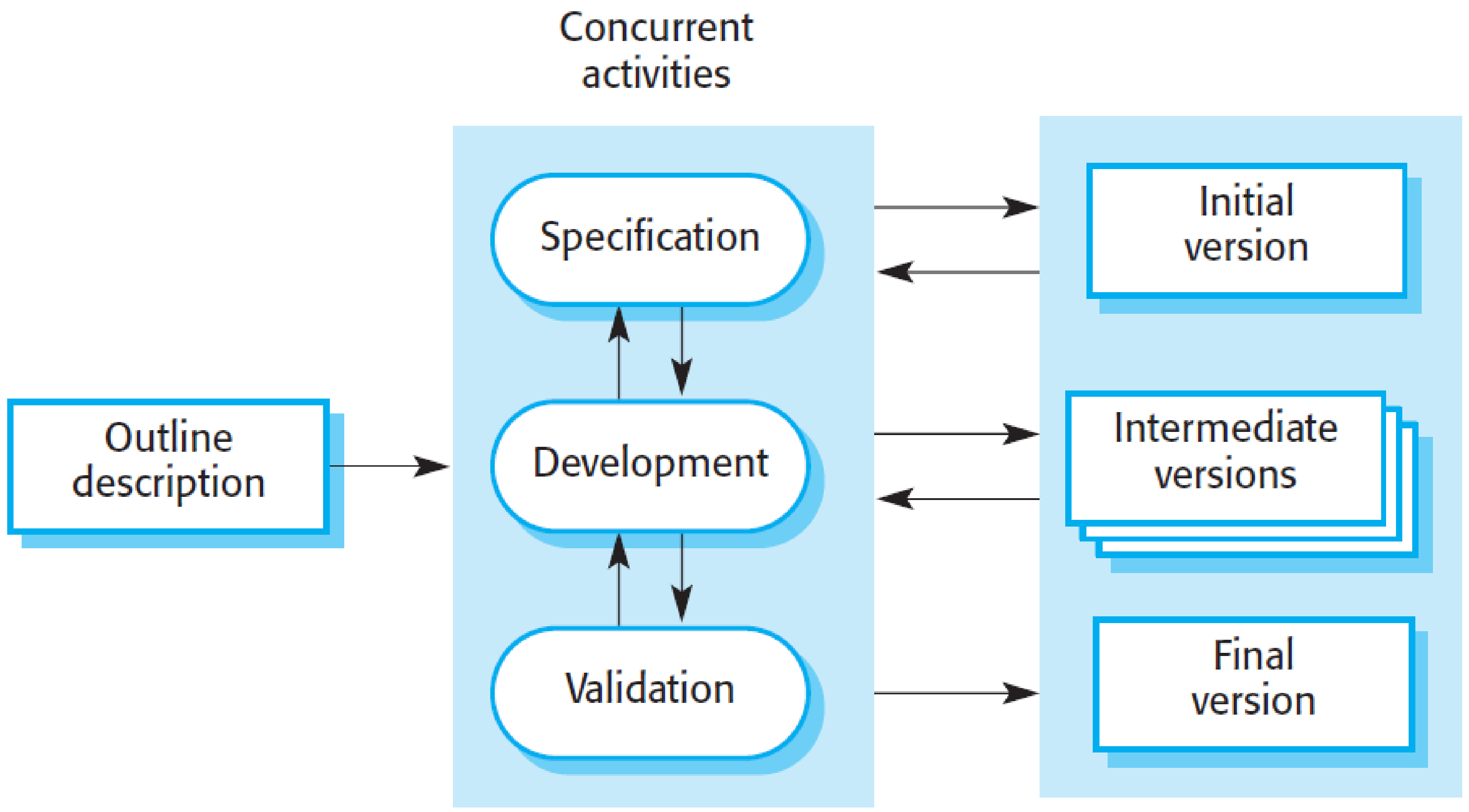
Incremental Model



Incremental Development

- Incremental development is based on the idea of developing an initial implementation, getting feedback/acceptance from users and evolving the software through several versions until the required system has been developed.
- System development is broken down into many mini development projects
  - Partial systems are successively built to produce a final total system
- Highest priority requirement is tackled first

Incremental Development Process



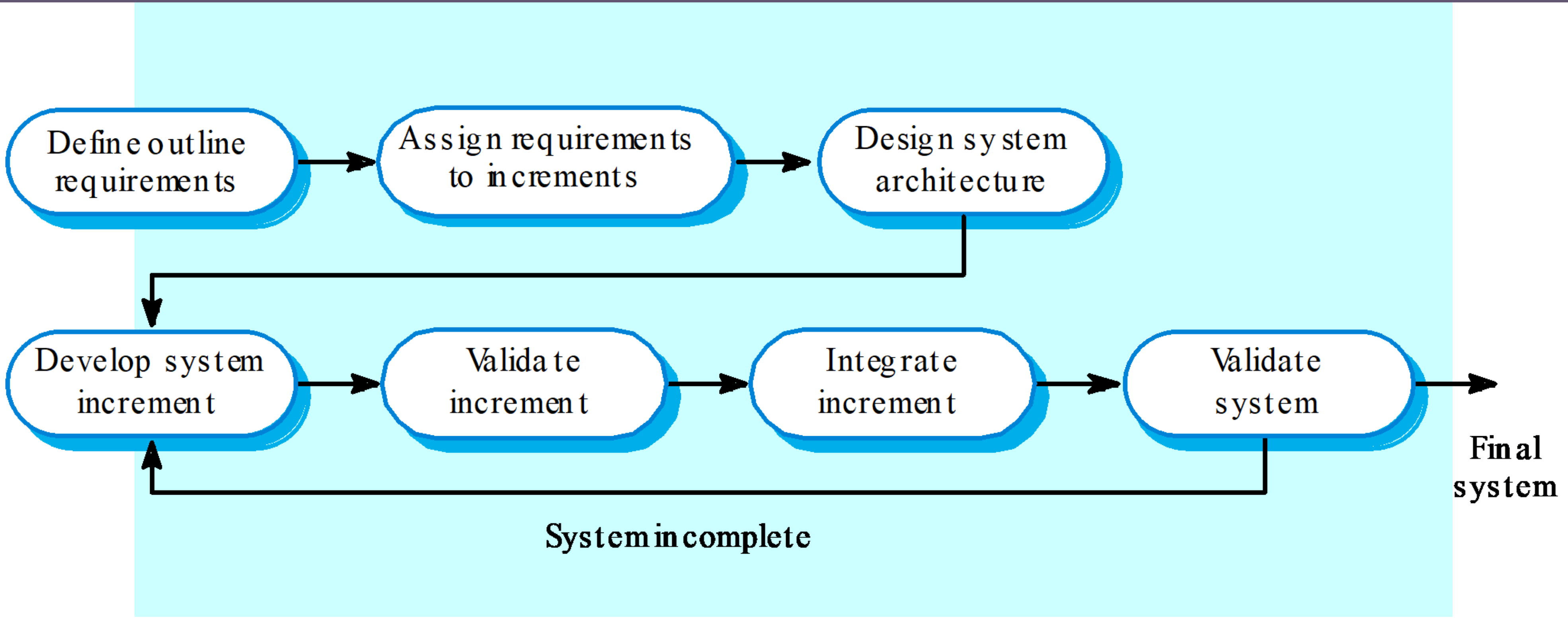
Incremental Development

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 model



Details - Incremental delivery

- During development, further requirements analysis for later increments can take place, but requirements changes for the current increment are not accepted.
- Once an increment is completed and delivered, it is installed in the customer's normal working environment.
  - They can experiment with the system, and this helps them clarify their requirements for later system increments.
- As new increments are completed, they are integrated with existing increments so that system functionality improves with each delivered increment.

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 Development Benefits

- The software will be generated quickly during the software life cycle
- It is flexible and less expensive to change requirements and scope
- Through out the development stages, some changes can be done
- This model is less costly compared to others
- A customer can respond to each building
- Errors are easy to be identified

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.

Incremental Delivery Problems

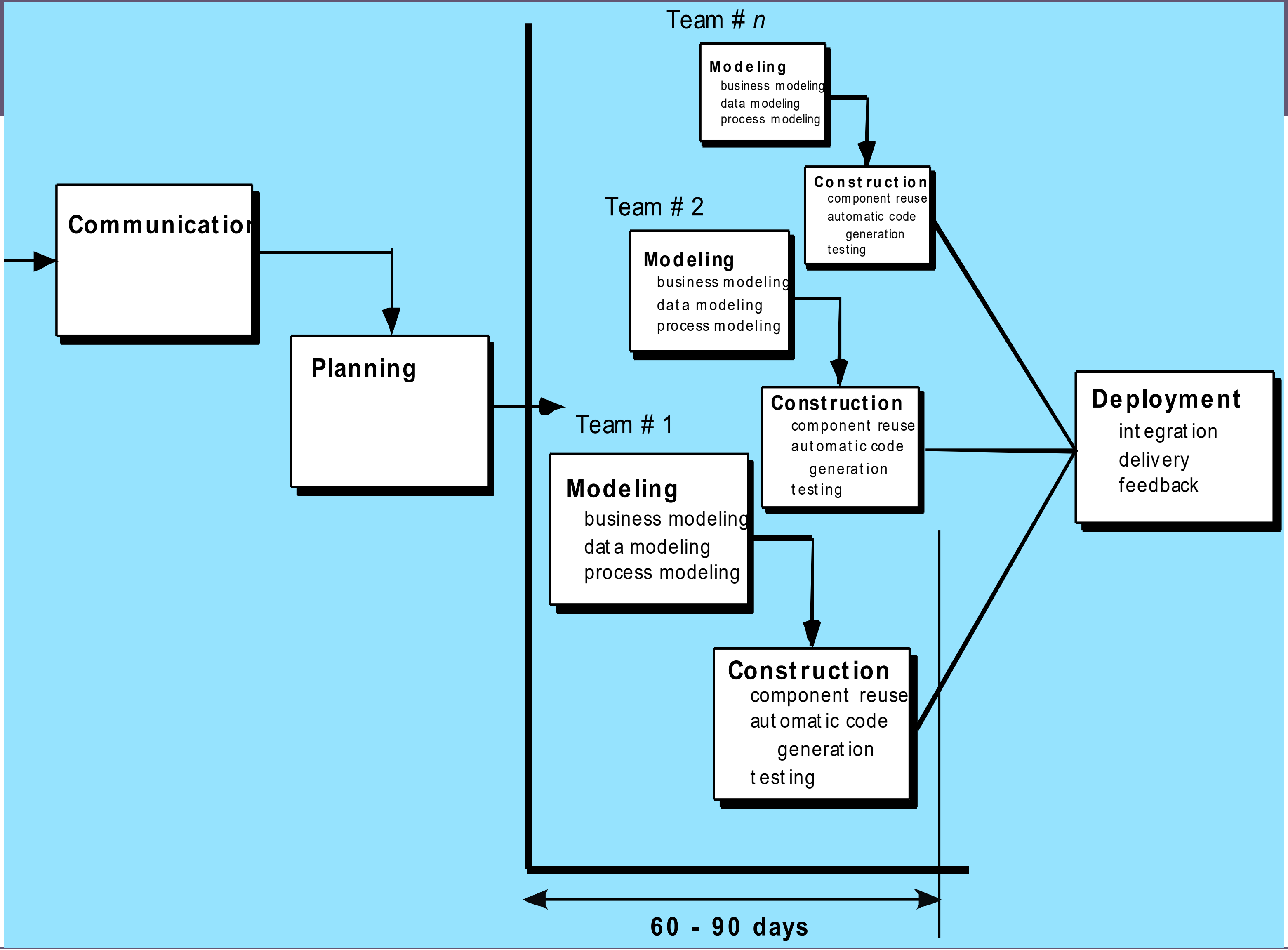
- Problematic when the new system is intended to replace an existing system.
  - Users need all of the functionality of the old system and are usually unwilling to experiment with an incomplete new system.
  - It is often impractical to use the old and the new systems alongside each other as they are likely to have different databases and user interfaces.
- 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 RAD Model

Rapid Application Development (RAD) is an incremental software development process model that emphasizes an extremely short development cycle.

If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a ‘fully functional system’ within very short time periods (eg. 60 to 90 days)

The RAD Model



Rapid Application Development (RAD)

- RAD is an **incremental software development process model**.
- It emphasizes **extremely short development cycles**, typically **60 to 90 days**.

Best suited for projects where:

- **Requirements are well-understood**, and
- **Scope is clearly constrained**.

Key Phases in RAD

1. Business Modeling
- Identify **information flow** between key business functions.

2. Data Modeling
- Use business information to define essential data objects.

3. Process Modeling
- Transform data objects to support the required business processes.

4. Application Generation
- Use 4GL tools or visual development tools to create applications.
- Emphasizes reuse of components.

5. Testing and Turnover
- All new components and their interfaces must be thoroughly tested.
- Final system is delivered for deployment.

Advantages and Challenges of RAD

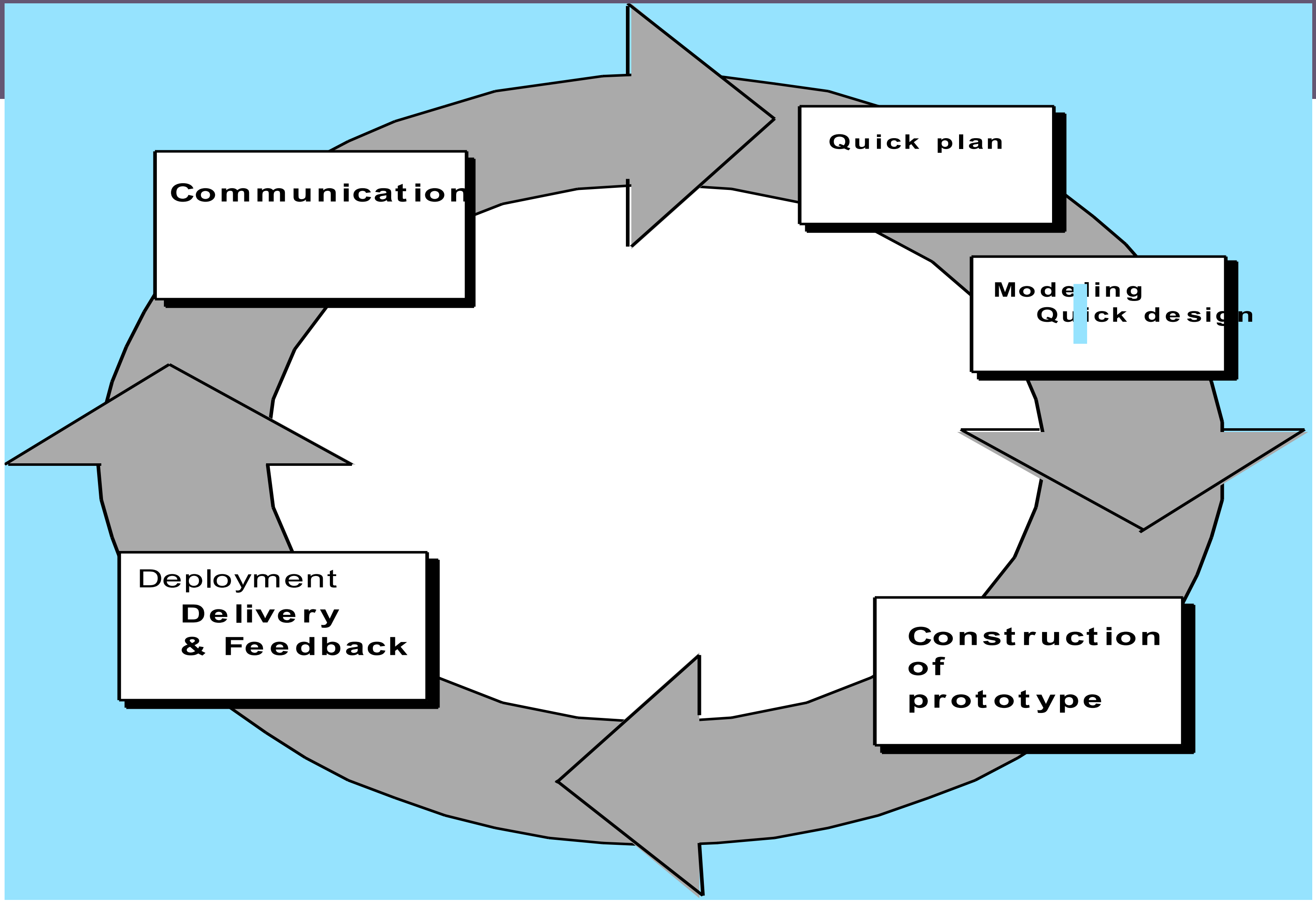
Advantages:

•**Fast delivery** of functional systems.
•**User involvement** from early stages.
•Promotes **component reuse**.
•**Reduces development time** significantly for suitable projects.

Challenges:

•Requires **sufficient skilled human resources** to create multiple RAD teams.
•Demands **high commitment** from developers and end-users.
•RAD is not suitable if:
•**Technical risks** are high (e.g., new technologies or complex integrations).
•The system **cannot be modularized** properly.
•There's a **need for high interoperability** with legacy systems.

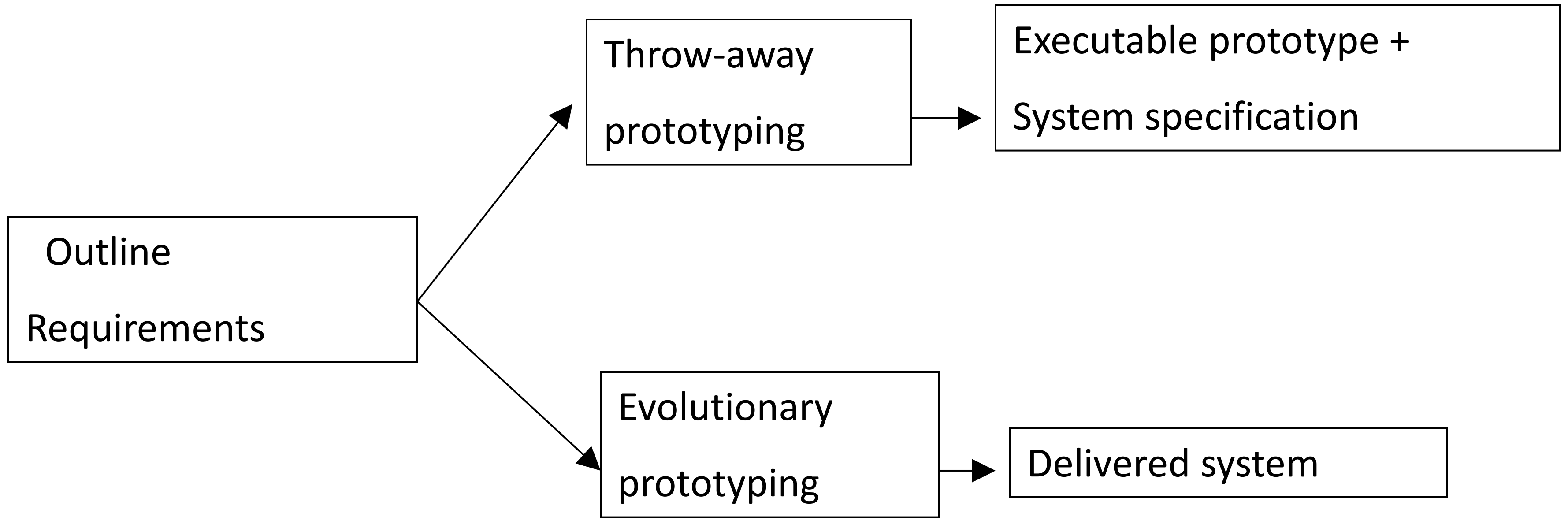
Evolutionary Models:



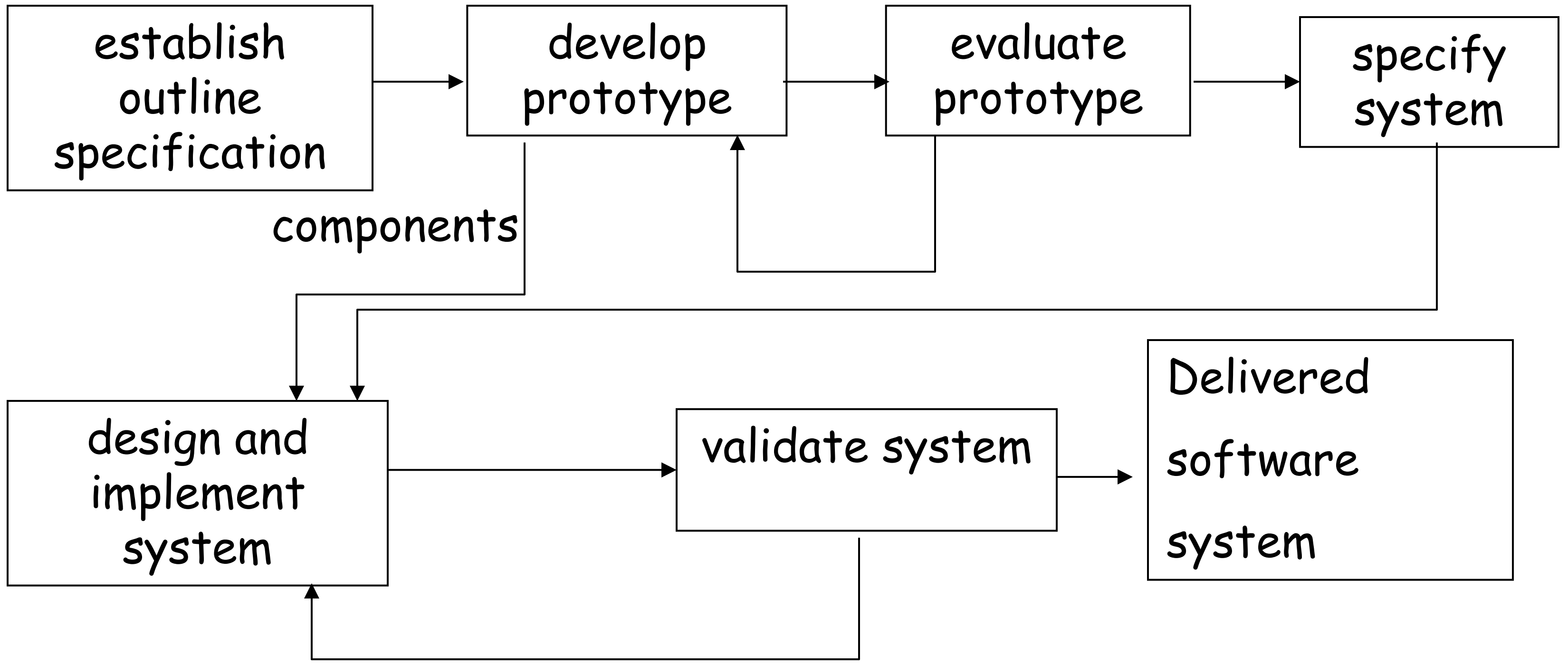
Prototyping

- A prototype is a simplified version of the final system.
  - Demonstrate core concepts
  - Explore design alternatives
  - Clarify requirements
  - Engage users early in the development process
- Types of Prototyping Techniques
  - Throw-away Prototyping
  - Evolutionary Prototyping

Throw-away and Evolutionary Prototyping



Throw-away Prototyping


establish outline specification
develop prototype
evaluate prototype
specify system
components
design and implement system
validate system
Delivered software system

Throw-away Prototyping

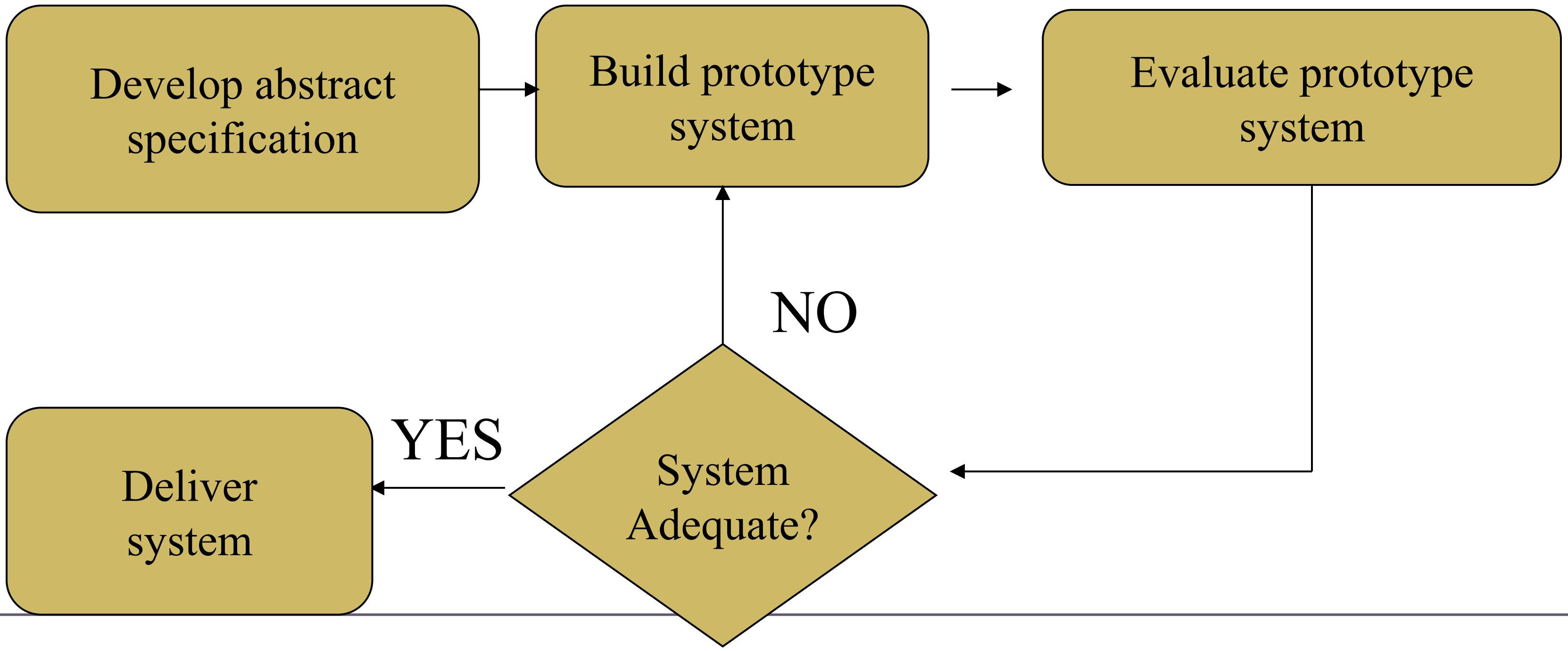
- To **clarify and understand system requirements** before actual development begins.
- Key Features
  - Starts with poorly understood or uncertain requirements.
  - A prototype is quickly built to explore and refine those requirements.
  - Once requirements are clarified, the prototype is discarded.
  - The actual system is developed from scratch, based on the refined understanding.

Challenges of Throw-away Prototyping

- **Incomplete Coverage**:
  Not all requirements—especially complex ones—can be effectively represented in a prototype.
- **No Legal Standing**:
  A prototype is **not a formal agreement**; it cannot be treated as a contract between stakeholders.
- **Non-functional Requirements Overlooked**:
  Aspects such as **performance, reliability, robustness, and safety** are often **not tested** in early prototypes.

Evolutionary Prototyping

- Begin with a basic initial implementation of the system.
- Present the prototype to users early for feedback.
- Use feedback to iteratively refine and improve the system.
- Continue evolving the prototype through multiple versions until it satisfies all user needs and becomes the final system.



Evolutionary Prototyping **Advantages**

* Effort of the prototype is not wasted

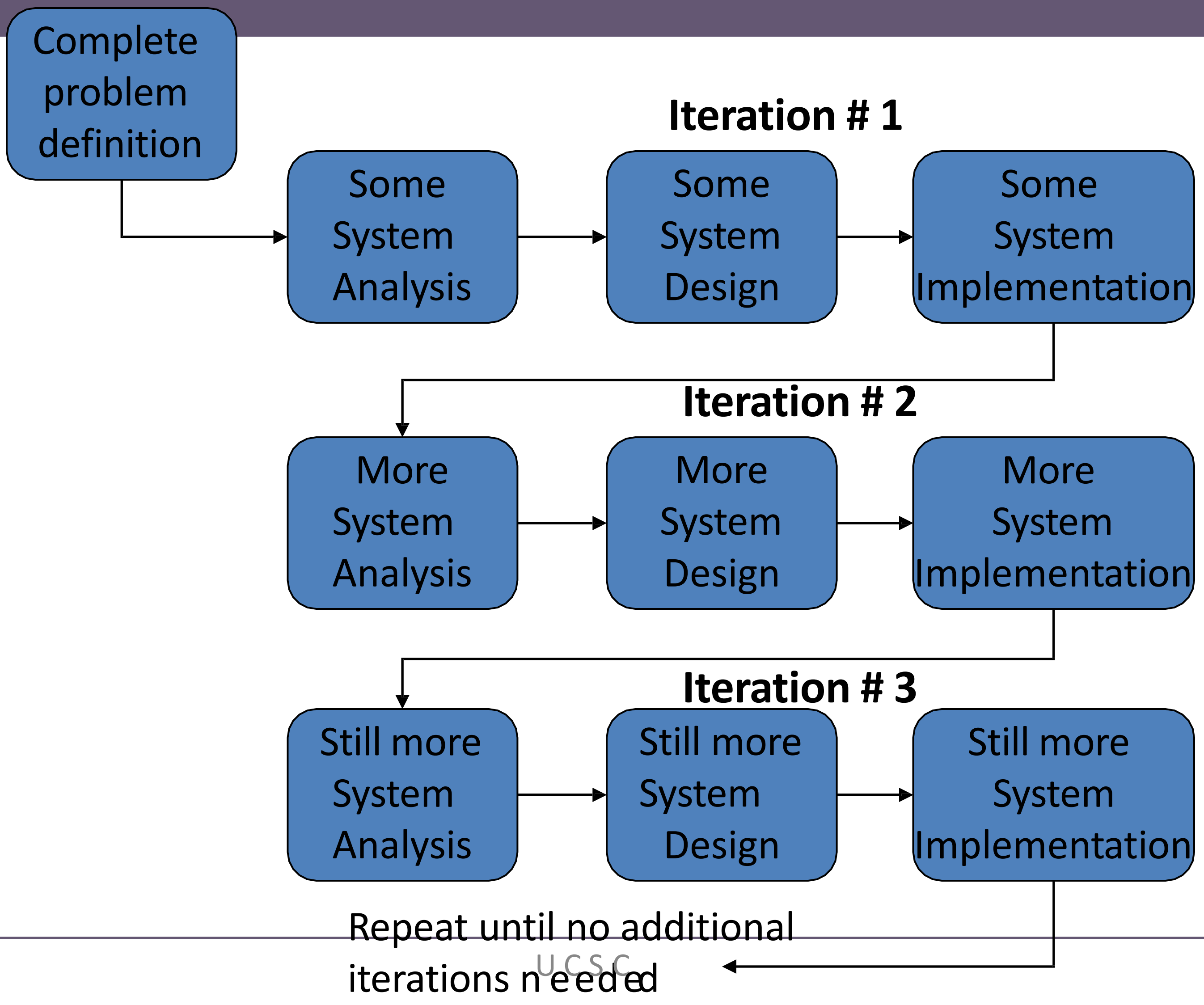
* Faster than the Waterfall model

* High level of user involvement from the start

* Technical or other problems discovered early – risk reduced

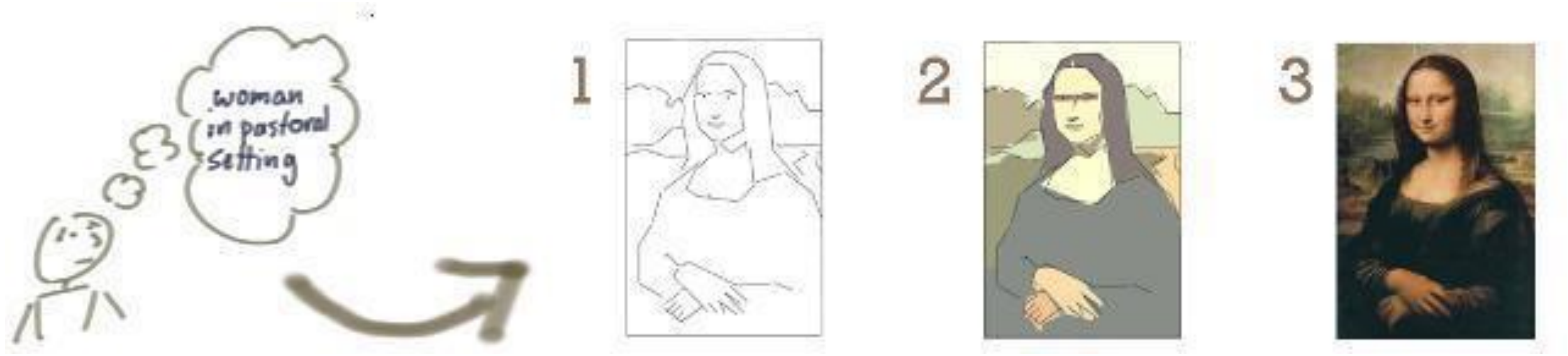
* mainly suitable for projects with vague and unstable requirements.

Iterative development

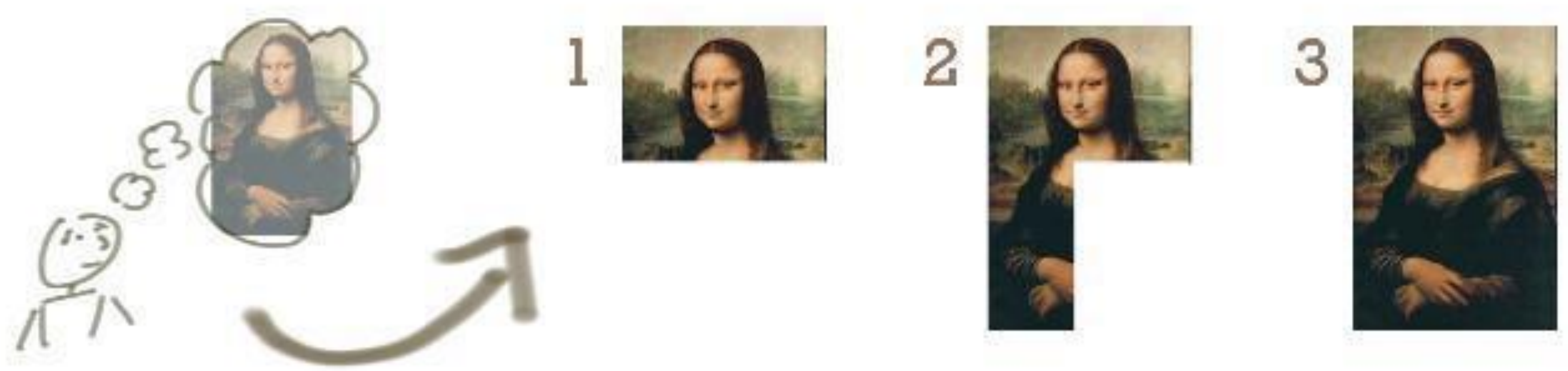

Complete problem definition
Iteration # 1
Some System Analysis
Some System Design
Some System Implementation
Iteration # 2
More System Analysis
More System Design
More System Implementation
Iteration # 3
Still more System Analysis
Still more System Design
Still more System Implementation
Repeat until no additional iterations needed

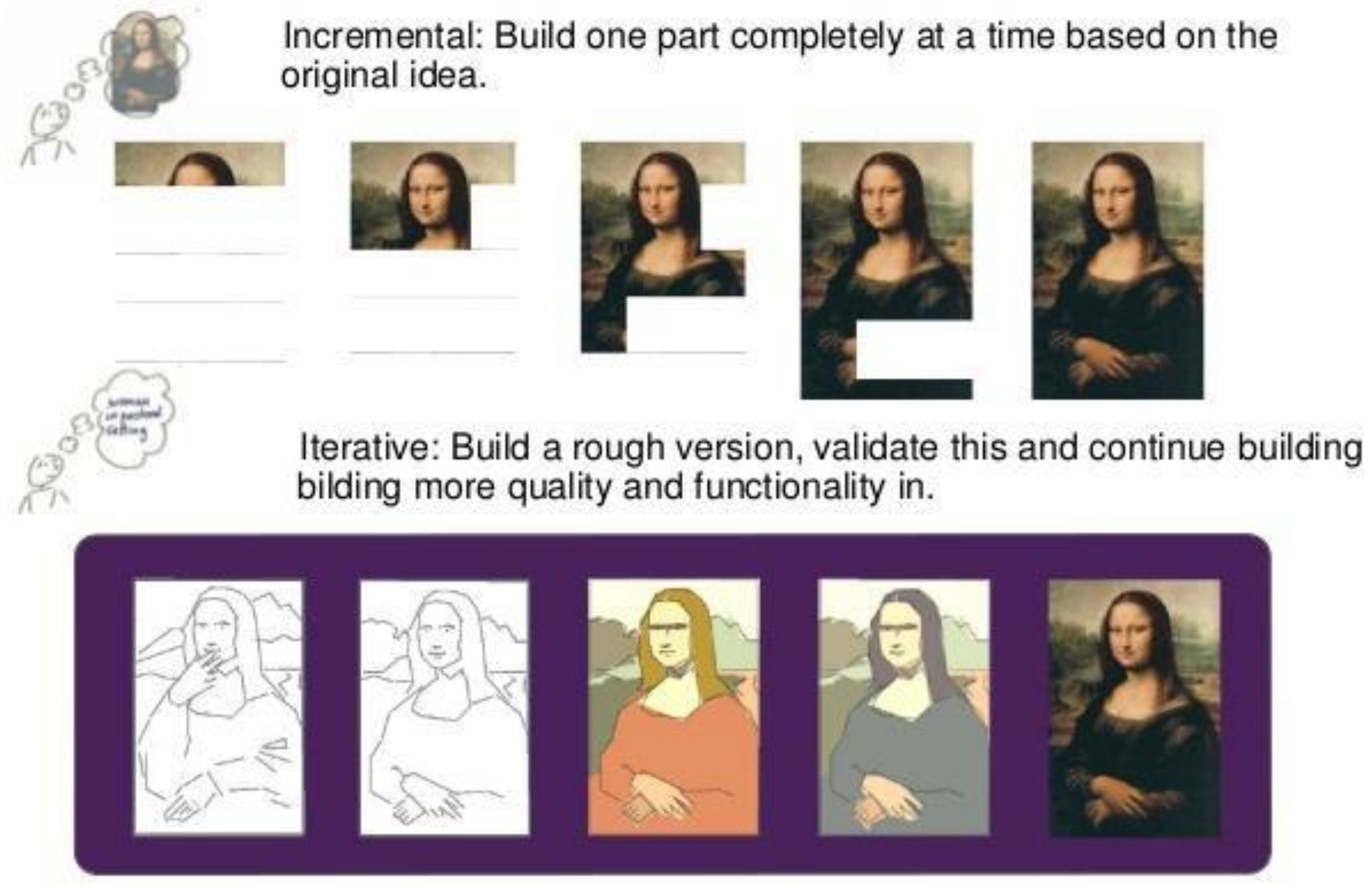
Incremental vs Iterative

Iterative



Incremental





Incremental: Build one part completely at a time based on the original idea.

Iterative: Build a rough version, validate this and continue building bilding more quality and functionality in.

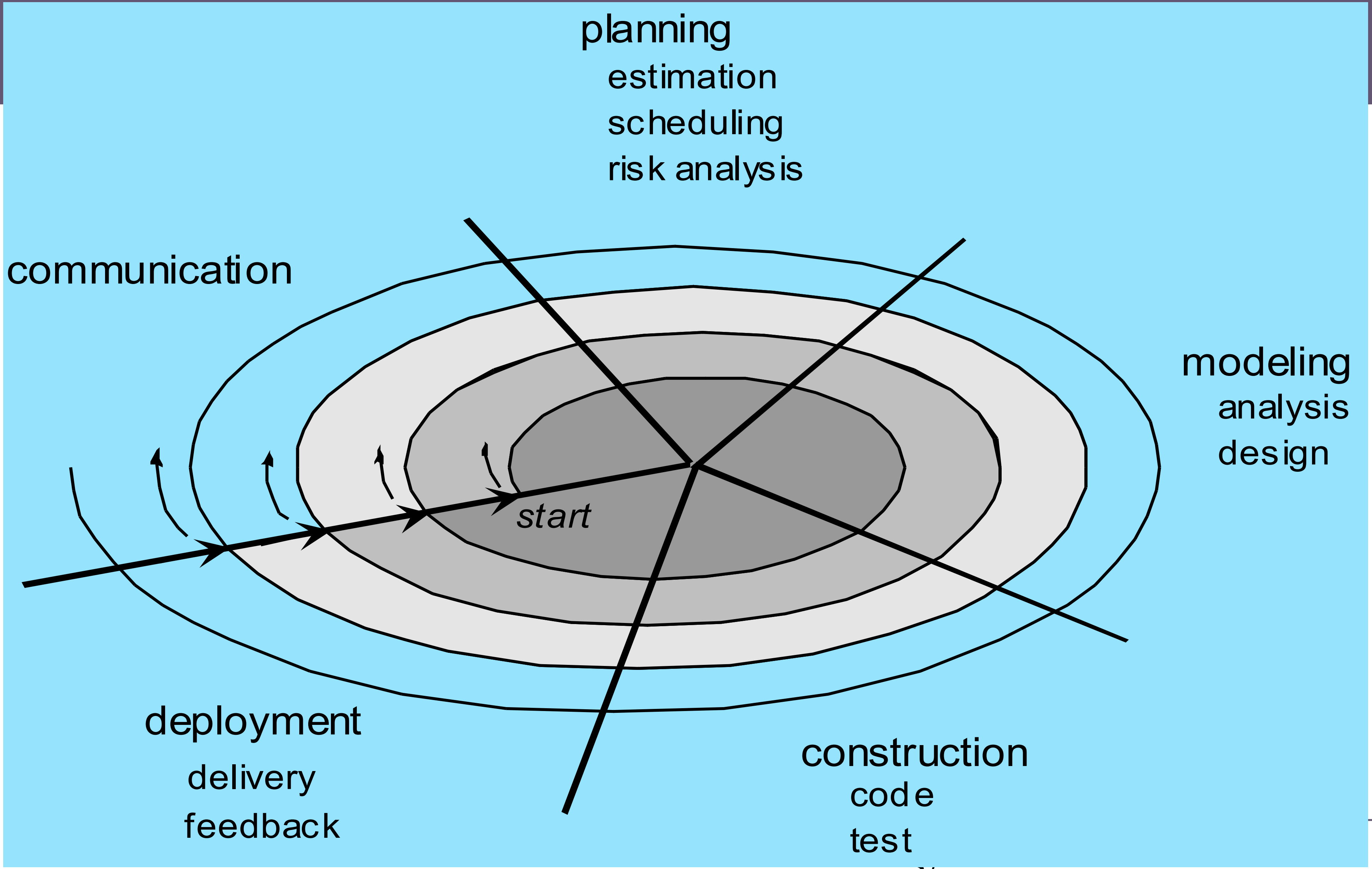
The Spiral Model

- This model is an evolutionary software process model that couples **the iterative nature of prototyping** with the controlled and systematic aspects of the linear sequential model.
- Using the spiral model software is developed in a series of incremental releases. During early iterations, the incremental release might be a paper model or prototype.

The spiral model is divided into four main task regions

- Determine goals, alternatives, constraints
- Evaluate alternatives and risks
- Develop and test
- Plan

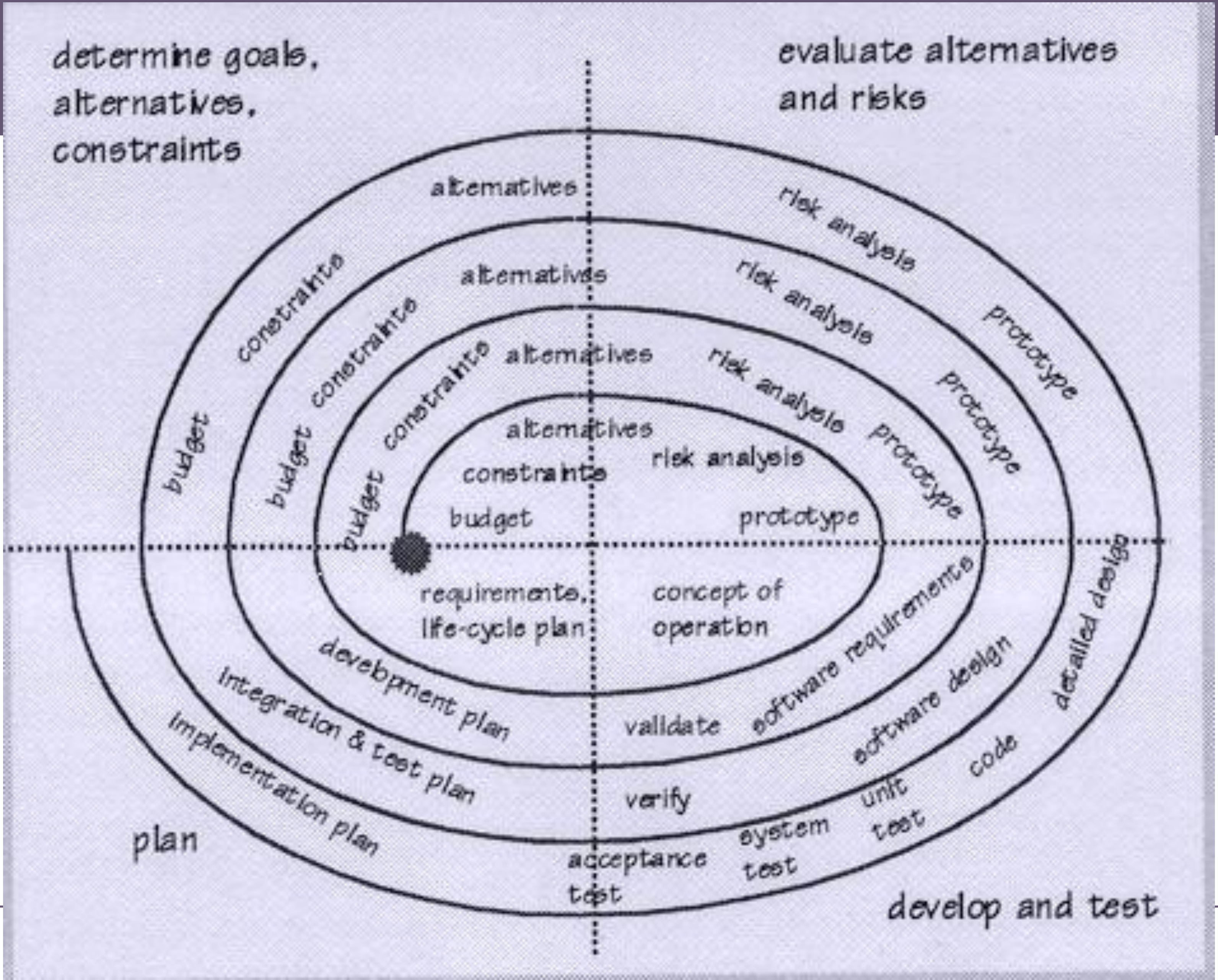
Evolutionary Models: The Spiral



Spiral Model of Software Development

- The process is visualized as a **spiral**, not a linear sequence of steps.
- Each **loop** in the spiral represents a **phase** of the software lifecycle (e.g., planning, risk analysis, development).
- There are **no fixed phases** like “specification” or “design.”
  - Instead, activities are selected **based on the specific project needs**.
- Promotes **iterative refinement** of the system with **customer feedback** at every stage.

Spiral Model

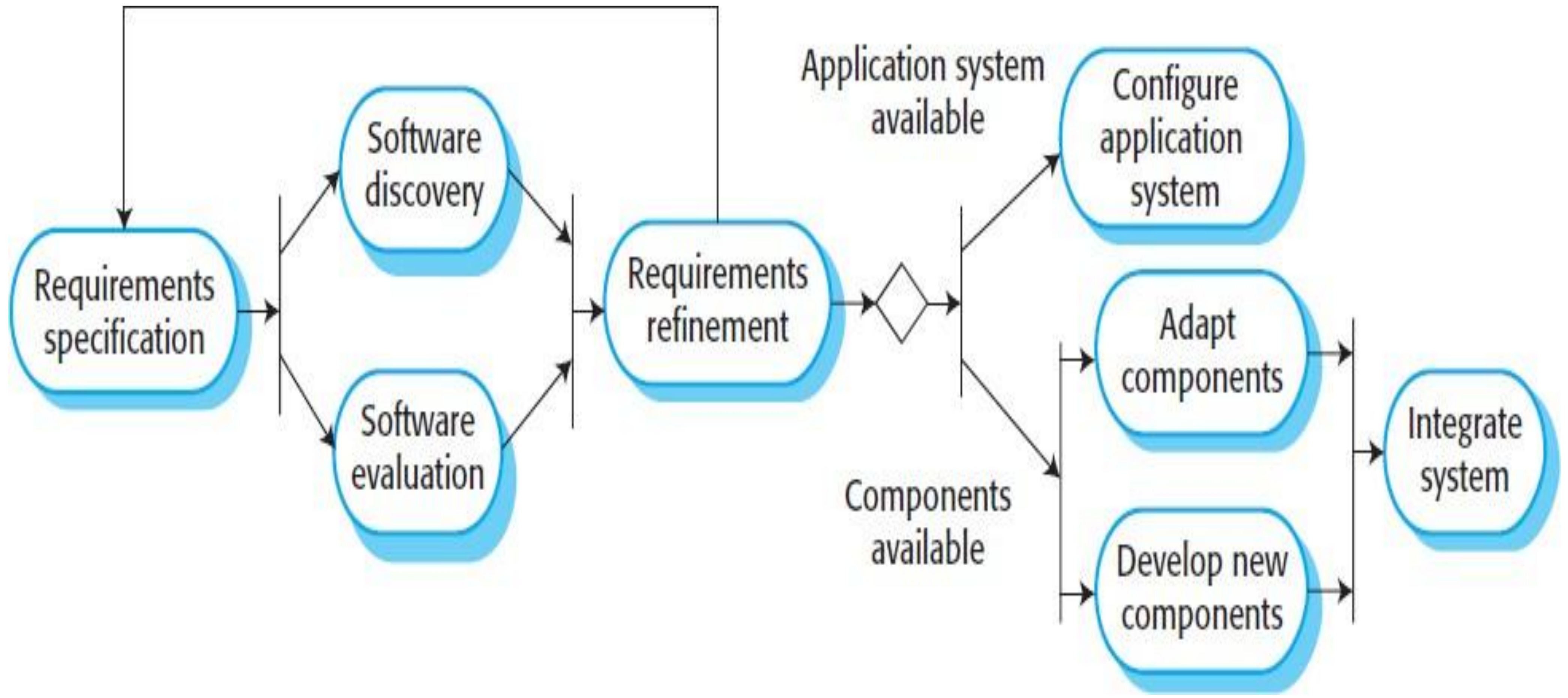


Reuse-Oriented Software Engineering

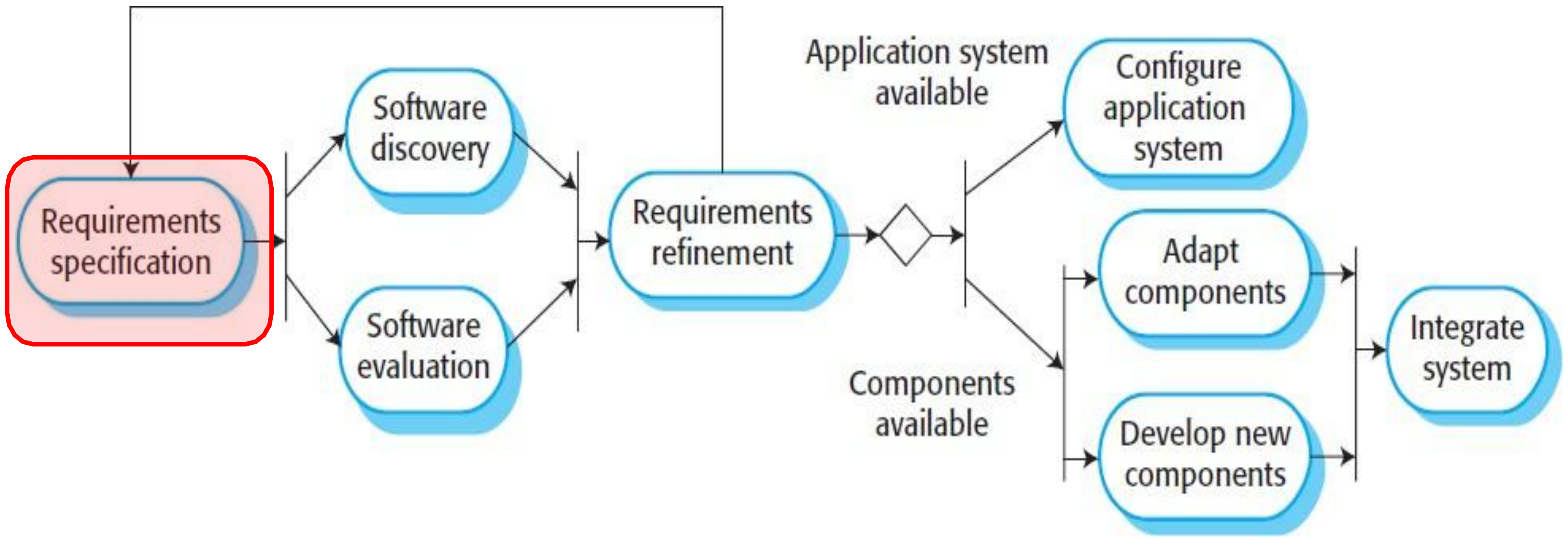
Integration and Configuration

@Eagle Eye

Process Stages of Reuse-Based Development
Component-based Development



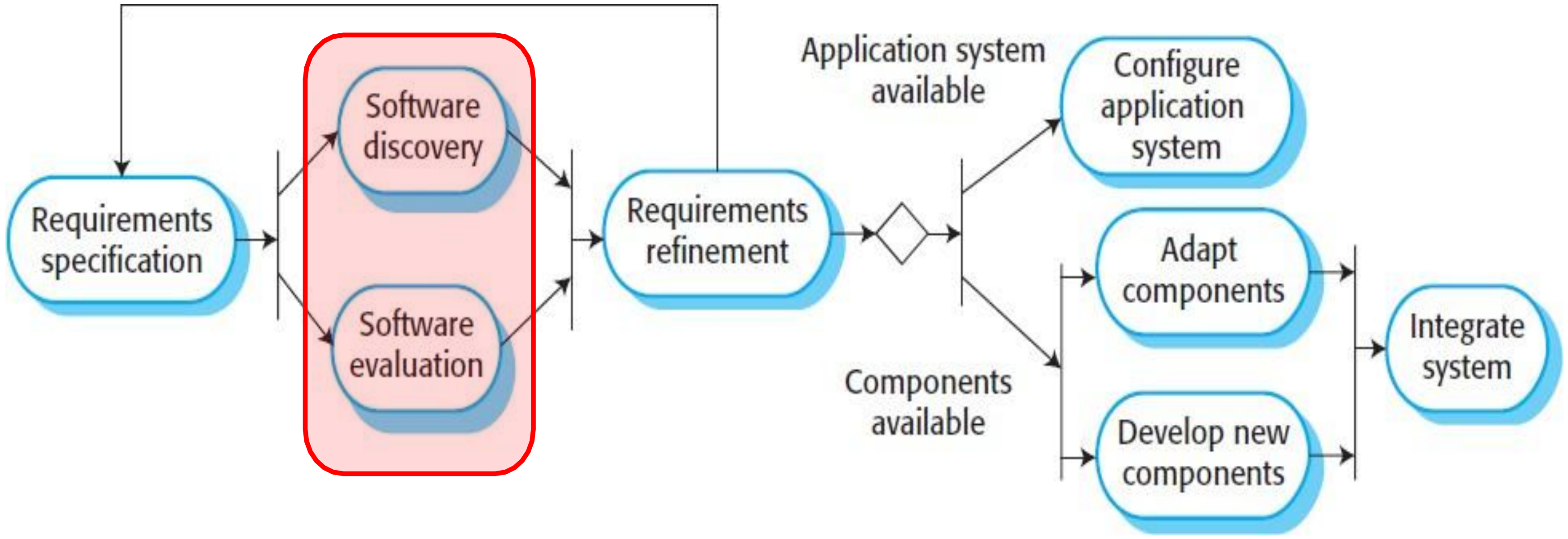
Process Stages of Reuse-Based Development



✧ Requirement specification;

- initial requirements for the system are proposed.
- These do not have to be elaborated in detail but should include brief descriptions of essential requirements and desirable system features.

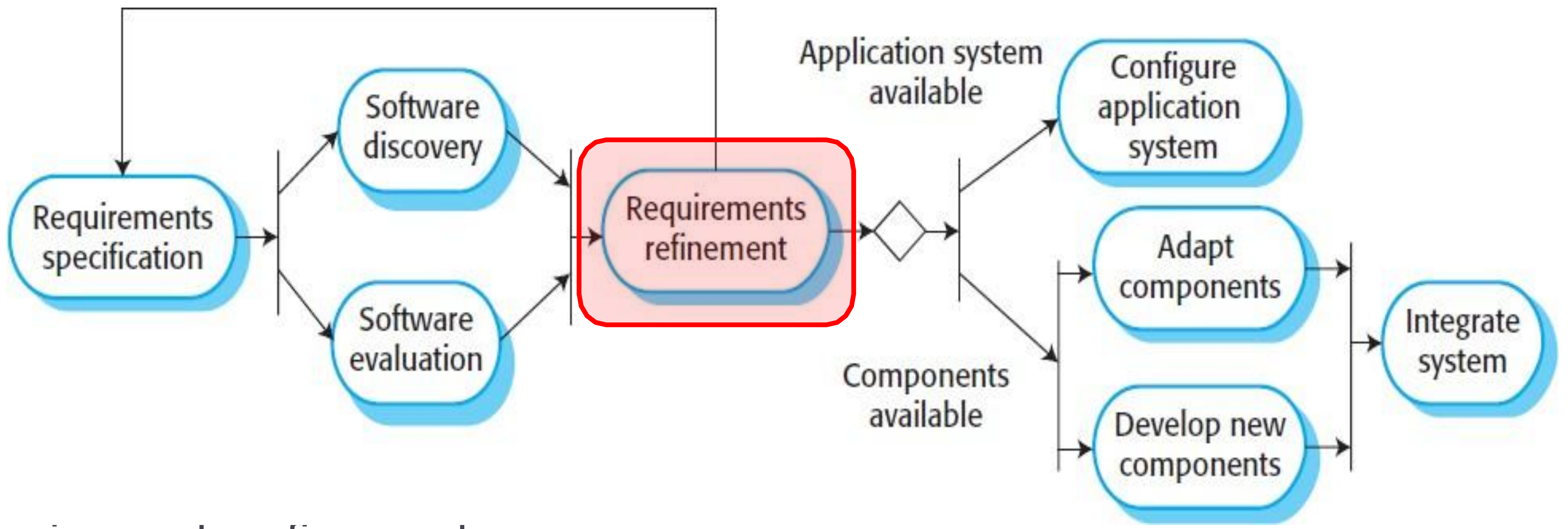
Process Stages of Reuse-Based Development



✧ Software Discovery and Evaluation

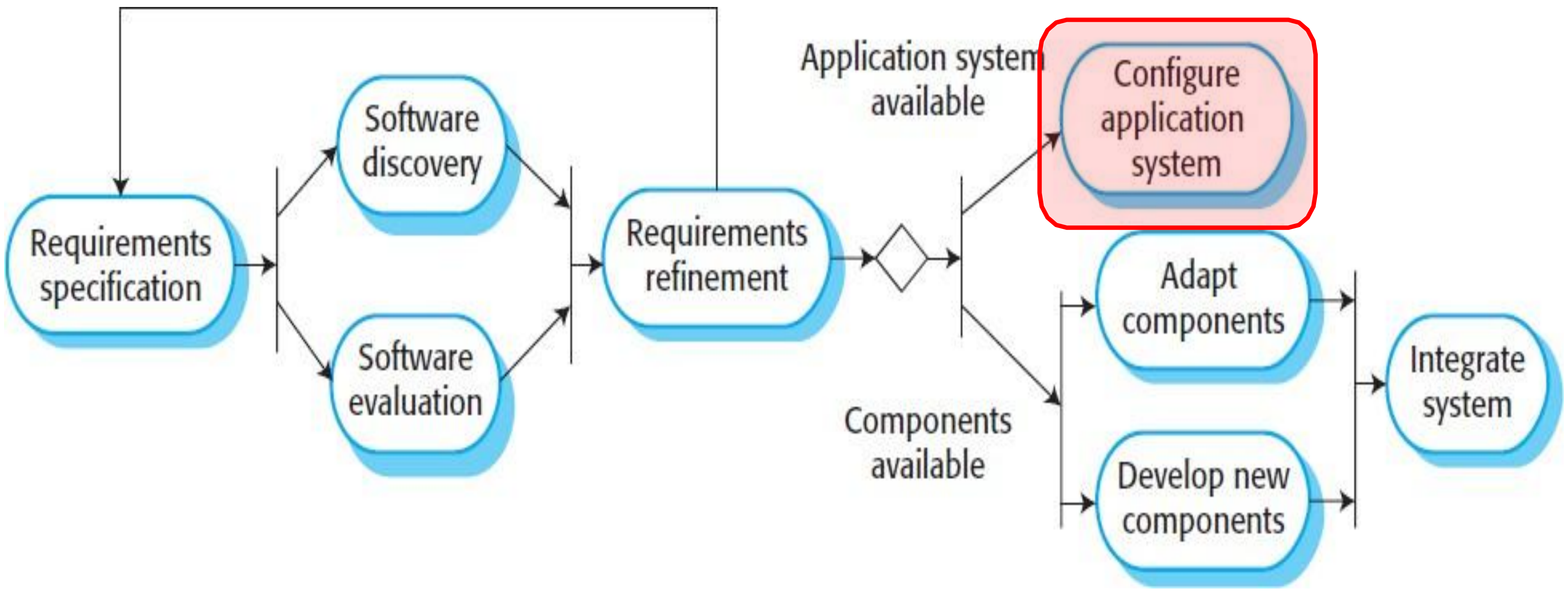
- Given an outline of the software requirements, a search is made for components and systems that provide the functionality required.
- Candidate components and systems are evaluated to see if they meet the essential requirements and if they are generally suitable for use in the system

Process Stages of Reuse-Based Development



- ✧ Requirements refinement:
  - During this stage, the requirements are refined using information about the reusable components and applications that have been discovered.
  - The requirements are modified to reflect the available components, and the system specification is re-defined. Where modifications are impossible, the component analysis activity may be reentered to search for alternative solutions.

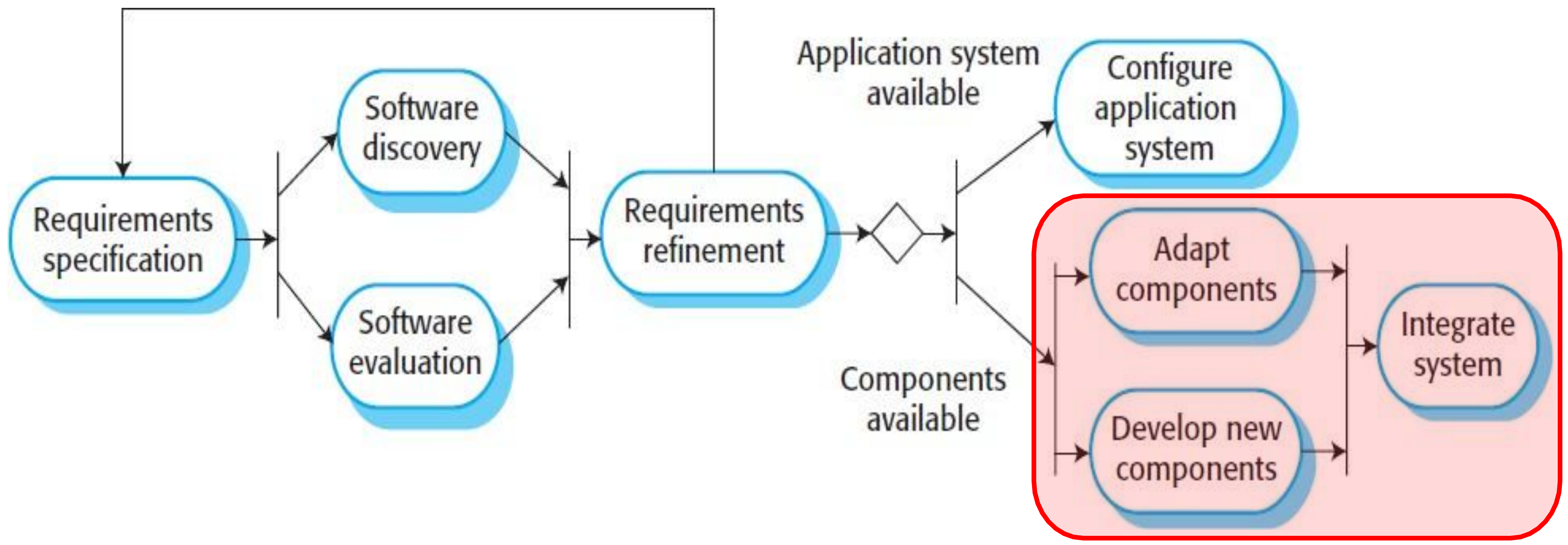
Process Stages of Reuse-Based Development



✧ Application system configuration:

- If an off-the-shelf application system that meets the requirements is available, it may then be configured for use to create the new system.

Process Stages of Reuse-Based Development



✧ Component adaptation and integration:

- If there is no off-the-shelf system, individual reusable components may be modified and new components developed.
- These are then integrated to create the system.

Reminding Questions

- What is meant by the term “software crisis”?
  - A. Instability of the development team to develop the software.
  - B. A situation in which experienced developers leave the company.
  - C. Inability of new software to inter-operate with existing software.
  - D. Inability to control the actual cost with respect to the budget over time.
  - E. Inability of new users to get familiar with the software.

Answer!

- What is meant by the term “software crisis”?
  A. Instability of the development team to develop the software.
  B. A situation in which experienced developers leave the company.
  C. Inability of new software to inter-operate with existing software.
  D. <u>Inability to control the actual cost with respect to the budget over time.</u>
  E. Inability of new users to get familiar with the software.

Questions

- Which of the following statement(s) is/are true?

1. Real projects rarely follow the sequential flow that the Waterfall model proposes.
2. It is not difficult to accommodate change after the process is underway in the Waterfall model.
3. The Waterfall model has the difficulty of accommodating the natural uncertainty that exists at the beginning of many projects.
4. The Waterfall model is suitable for projects which have unclear and unstable requirements.
5. It is often very easy for the customer to state all requirements explicitly.

Answer

- Which of the following statement(s) is/are true?

1. **Real projects rarely follow the sequential flow that the Waterfall model proposes.**
2. It is not difficult to accommodate change after the process is underway in the Waterfall model.
3. **The Waterfall model has the difficulty of accommodating the natural uncertainty that exists at the beginning of many projects.**
4. The Waterfall model is suitable for projects which have unclear and unstable requirements.
5. It is often very easy for the customer to state all requirements explicitly.

Reuse-Oriented Software Engineering

- Questions
  - What are the advantages of using reuse-oriented software engineering approach for software development?

Reuse-Oriented Software Engineering

What are the advantages of using reuse-oriented software engineering approach for software development?

- reducing the amount of software to be developed
- reducing cost and risks
- usually leads to faster delivery of the software

Evolution Stages

Decade	**Model**	**Approach**
1970s	Waterfall	Incremental
1980s	Spiral	Iterative
1990s	Unified Process (UP)	Hybrid
2001s	Agile	Non-plan driven
2010s	DevOps	Tools based
2020s	AI-Based Process Models	Intelligent / Next-Gen

Plan-driven Vs. Agile Processes (Process Culture)

- Plan-driven processes- all of the process activities are planned in advance and progress is measured against this plan.
- 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.

Conventional Software Development Models (1)

Waterfall Model

- Sequential flow through requirement, design, implementation, testing, and maintenance phases.

V-Model (Verification and Validation Model)

- Extension of the waterfall with corresponding testing phases for each development stage.

Incremental Model

- Delivers system in parts (increments), but planning for all increments happens early.

Conventional Software Development Models (2)

Spiral Model

- Emphasizes risk management with iterative cycles, but each iteration is thoroughly planned.

Prototyping Models (Throw-away and Evolutionary)

- Early prototype created for understanding requirements, then discarded; followed by plan-driven development.
- Prototype becomes the system

Conventional Software Development Models (3)

Rapid Application Development (RAD)

- Uses predefined phases and reusable components; requires heavy initial planning and commitment

Reused-Oriented Software Development

- Assembles systems from existing components; design reuse and planned integration are key.

Formal Methods Model

- Uses mathematically-based specifications and verifications; highly structured and planned.

Road Map

1. Software process models
2. Process activities
3. Coping with change
4. Process Improvement

2. Process Activities

The four basic process activities:

a. Software Specification
b. Software Development
c. Software Validation
d. Software Evolution

Process Activities

The four basic process activities:

a. Software Specification
b. Software Development
c. Software Validation
d. Software Evolution

a. Software Specification (Requirements Engineering)

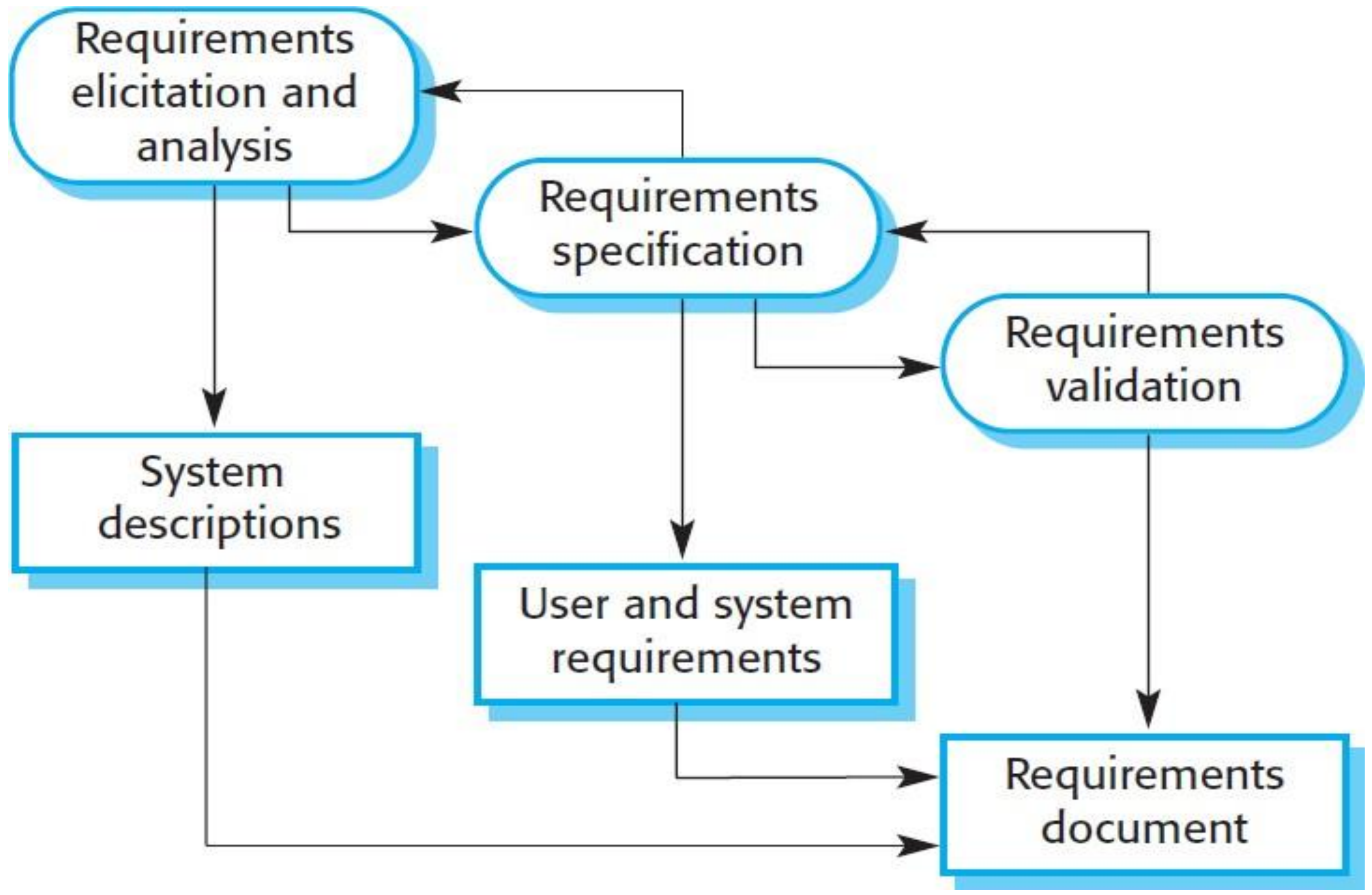
- The process of establishing what services are required and the constraints on the system's operation and development.
- Before the requirements engineering process starts, a company may carry out a feasibility or marketing study to assess whether or not there is a need or a market for the software and whether or not it is technically and financially realistic to develop the software required.

Software Specification (Requirements Engineering)

Requirements engineering process

- Requirements elicitation and analysis
  - What do the system stakeholders require or expect from the system?
- Requirements specification
  - Translate the information gathered during requirements analysis into a document that defines a set of requirements.
  - User requirements and System requirements
- 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

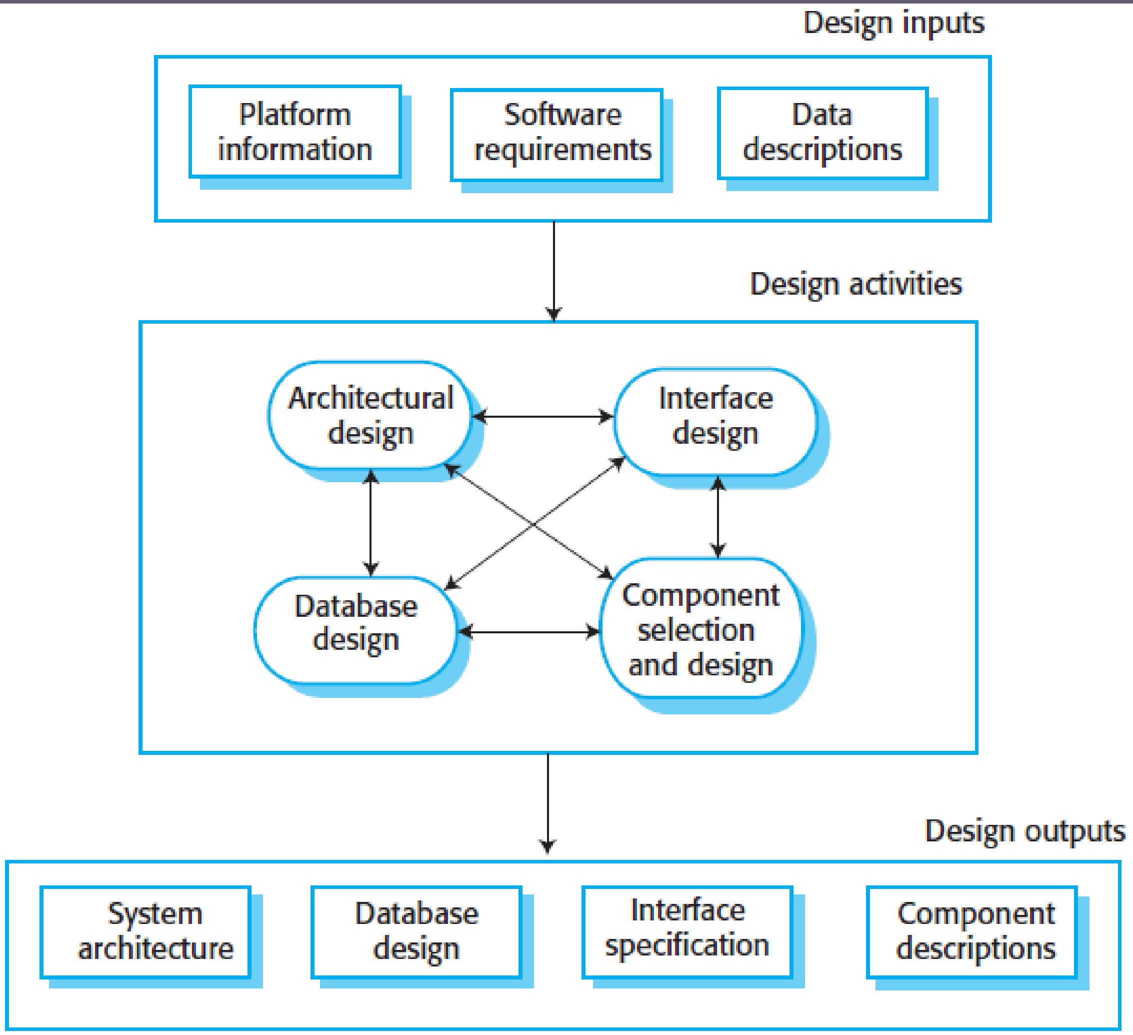
2. Process Activities

- The four basic process activities:
  a. Software Specification
  b. Software Development
  c. Software Validation
  d. Software Evolution

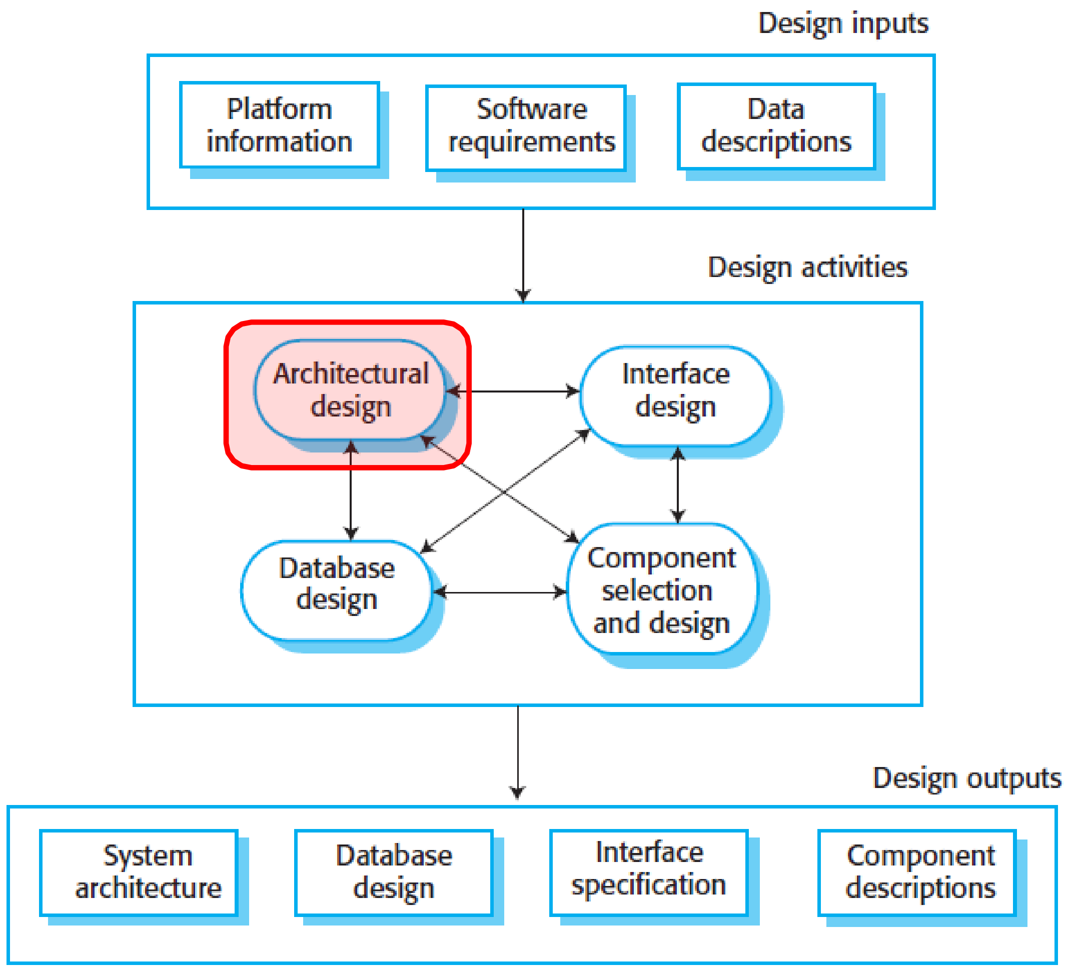
Software Design and Implementation

- 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

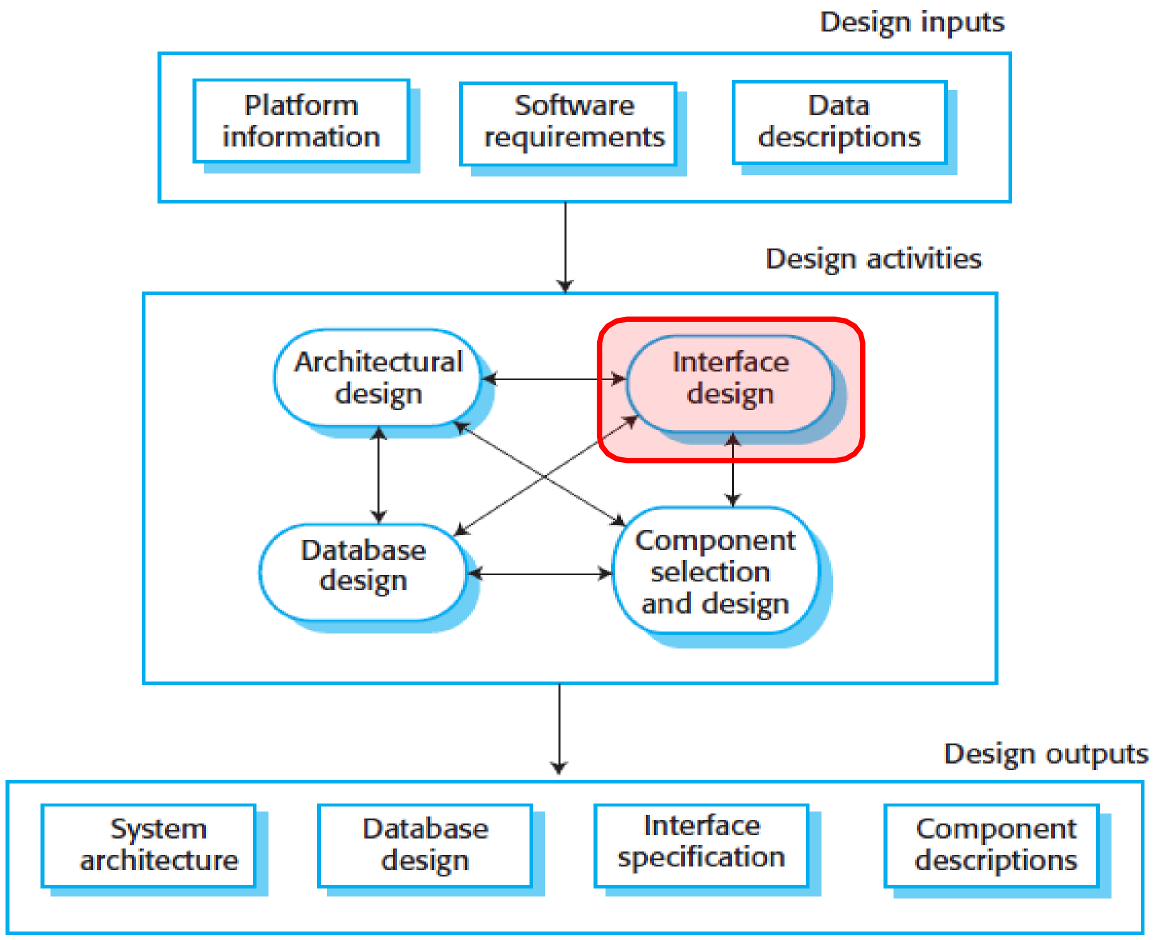


A General Model of the Design Process



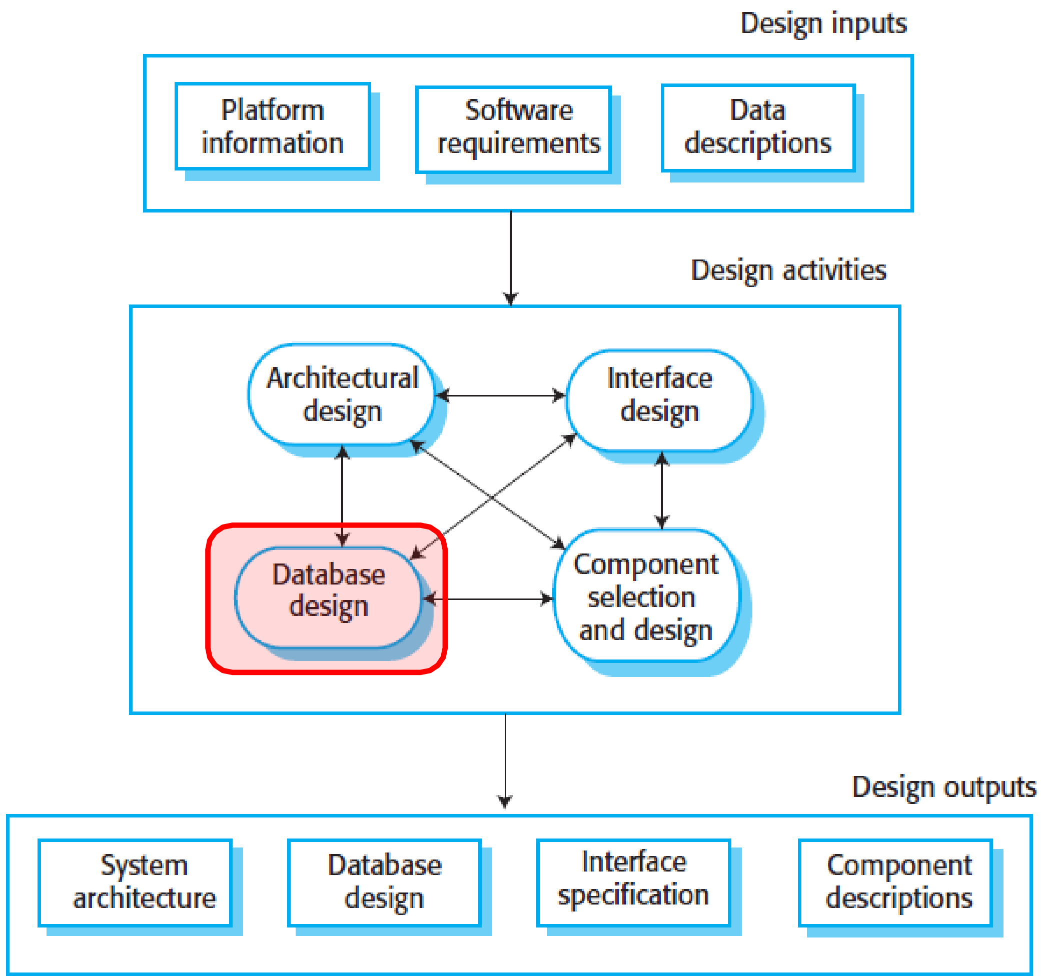
- *Architectural design:*
  - where you identify the overall structure of the system, the principal components (sometimes called sub- systems or modules), their relationships and how they are distributed.

A General Model of the Design Process



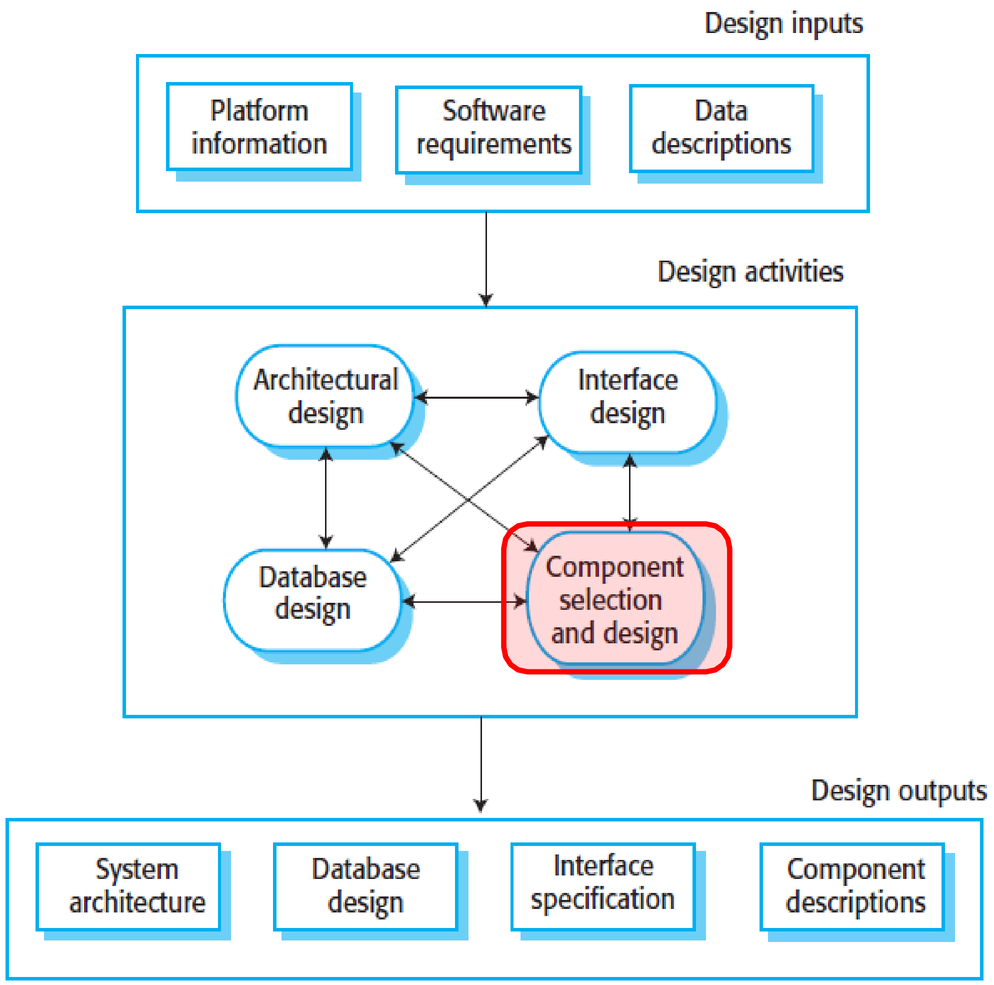
- *Interface design:*
  - where you define the interfaces between system components.

A General Model of the Design Process



- *Database design:*
  - – where you design the system data structures and how these are to be represented in a database

A General Model of the Design Process



- *Component selection and design:*
  - where you search for reusable components and, if no suitable components are available, design new software components

2. Process Activities

a. The four basic process activities:
   a. Software Specification
   b. Software Development
   c. Software Validation
   d. Software Evolution

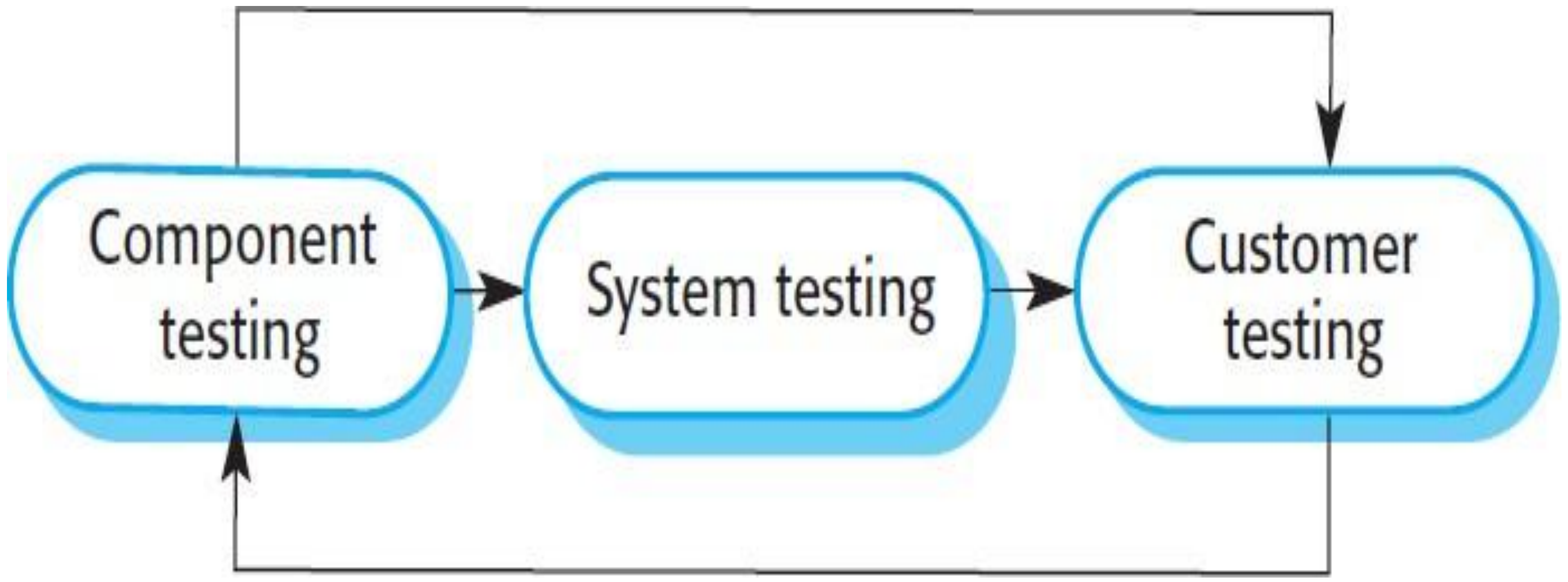
Software Validation

- 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.

Verification and Validation

- ***Verification*** *is the process of checking that the software meets the specification*
  - *Walkthrough*
  - *Inspection*
  - *Review*
- ***Validation*** *is the process of checking whether the specification captures the customer's needs.*
  - *Testing*

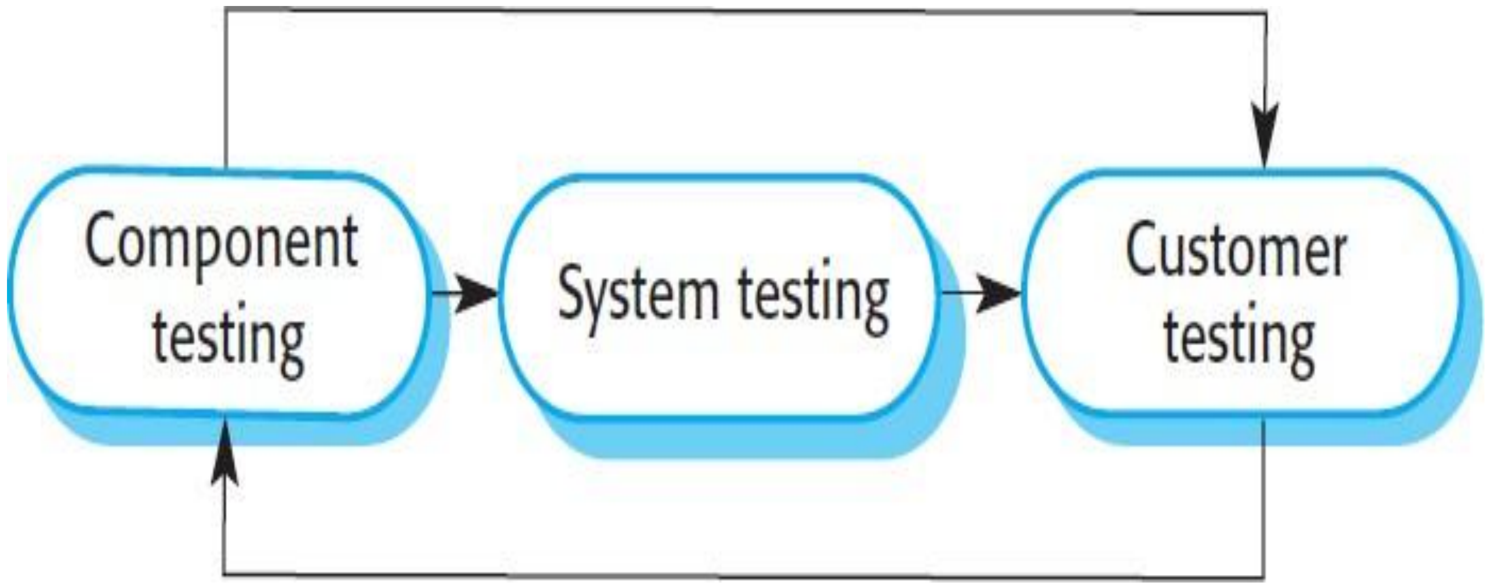
Stages of Testing



✧ Development or component testing

- Individual components are tested independently;
- Components may be functions or objects of these entities.

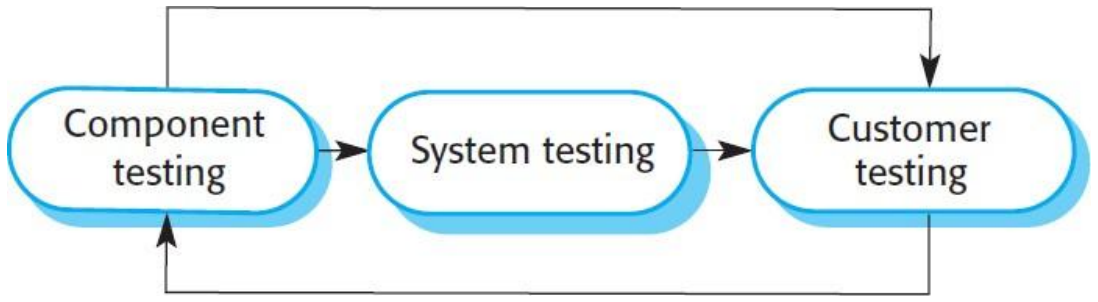
Stages of Testing



✧ System testing

- Testing of the system as a whole. Testing of emergent properties is particularly important.

Stages of Testing



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

Testing Phases in a plan-driven Software Process

- ✧When a plan-driven software process is used testing is driven by a set of test plans.
- ✧An independent team of testers carries out these test plans, which have been developed from the system specification and design.

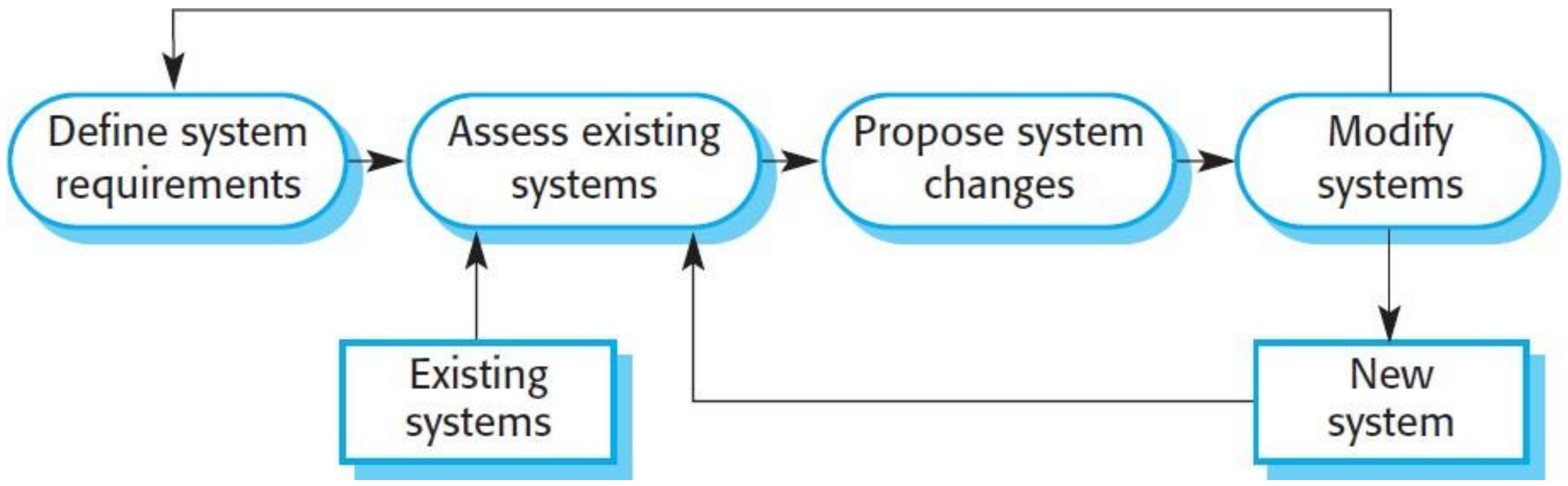
2. Process Activities

a. The four basic process activities:
   a. Software Specification
   b. Software Development
   c. Software Validation
   d. Software Evolution

Software Evolution (Maintenance)

- The flexibility of software is one of the main reasons why more and more software is being incorporated into large, complex systems.
- Once a decision has been made to manufacture hardware, it is very expensive to make changes to the hardware design.
- However, changes can be made to software at any time during or after the system development.
- Even extensive changes are still much cheaper than corresponding changes to system hardware.

System Evolution



Road Map

1. Software process models
2. Process activities
3. Coping with change
4. Process Improvement

Coping with change

- Change is unavoidable 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

Approaches to reduce the costs of rework

- Change anticipation, where the software process includes activities that can predict 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 and software are designed so that changes can be easily made to the system.
  - 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.

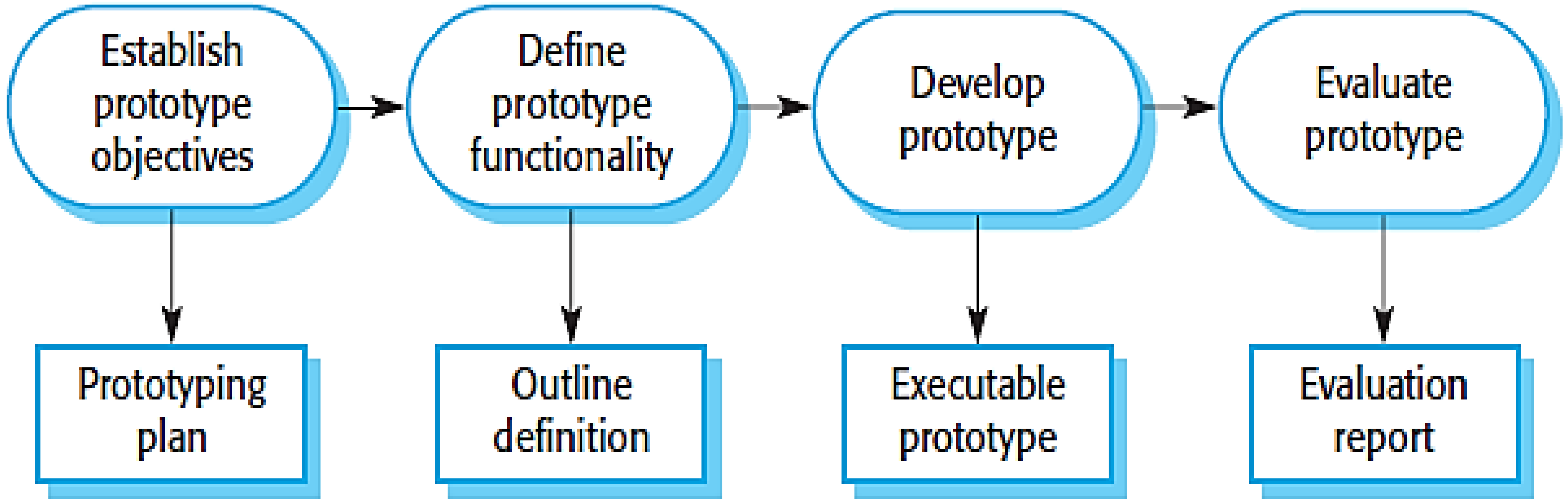
Software prototyping

- 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;
  - System design processes to explore software solutions and develop a UI design;

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



Road Map

1. Software process models
2. Process activities
3. Coping with change
4. Process Improvement

The software process improvement

- A Structured set of activities required

  eg. Specification, Design, Validation, Verification, Evolution
- Activities vary depending on the organisation and the type of system being developed
- Must be explicitly modelled if it is to be managed

Process characteristics

- - Understandability
- - Visibility
- - Acceptability
- - Reliability
- - Rapidity

- Robustness
- Maintainability
- Supportability

Process characteristics

- **Understandability**

To what extent is the process explicitly defined and how easy is it to understand the process definition?

- **Visibility**

Do the process activities culminate in clear results so that the progress of the process is externally visible?

- **Acceptability**

Is the defined process acceptable to and usable by the engineers responsible for producing the software project?

- **Reliability**

Is the process is designed in such a way that process errors are avoided or trapped before they result in product errors?

- **Rapidity**
- How fast can the process of delivering a system from a given specification be completed?
- **Robustness**

- Can the process continue in spite of unexpected problems?

- **Maintainability**
- Can the process evolve to reflect changing organizational requirements or identified process improvements?

Supportability

- To what extent can the process activities be supported by CASE tools?

Questions

- **The prototyping model of software development is well suited;**
    1. When requirements are well defined
    2. For projects with large development teams
    3. When a customer cannot define requirements clearly.

Key points – Summary

- 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 the 'waterfall' model, incremental development, and reuse- oriented development.

Key points - summary

- 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.