TICT3153
SOFTWARE ENGINEERING

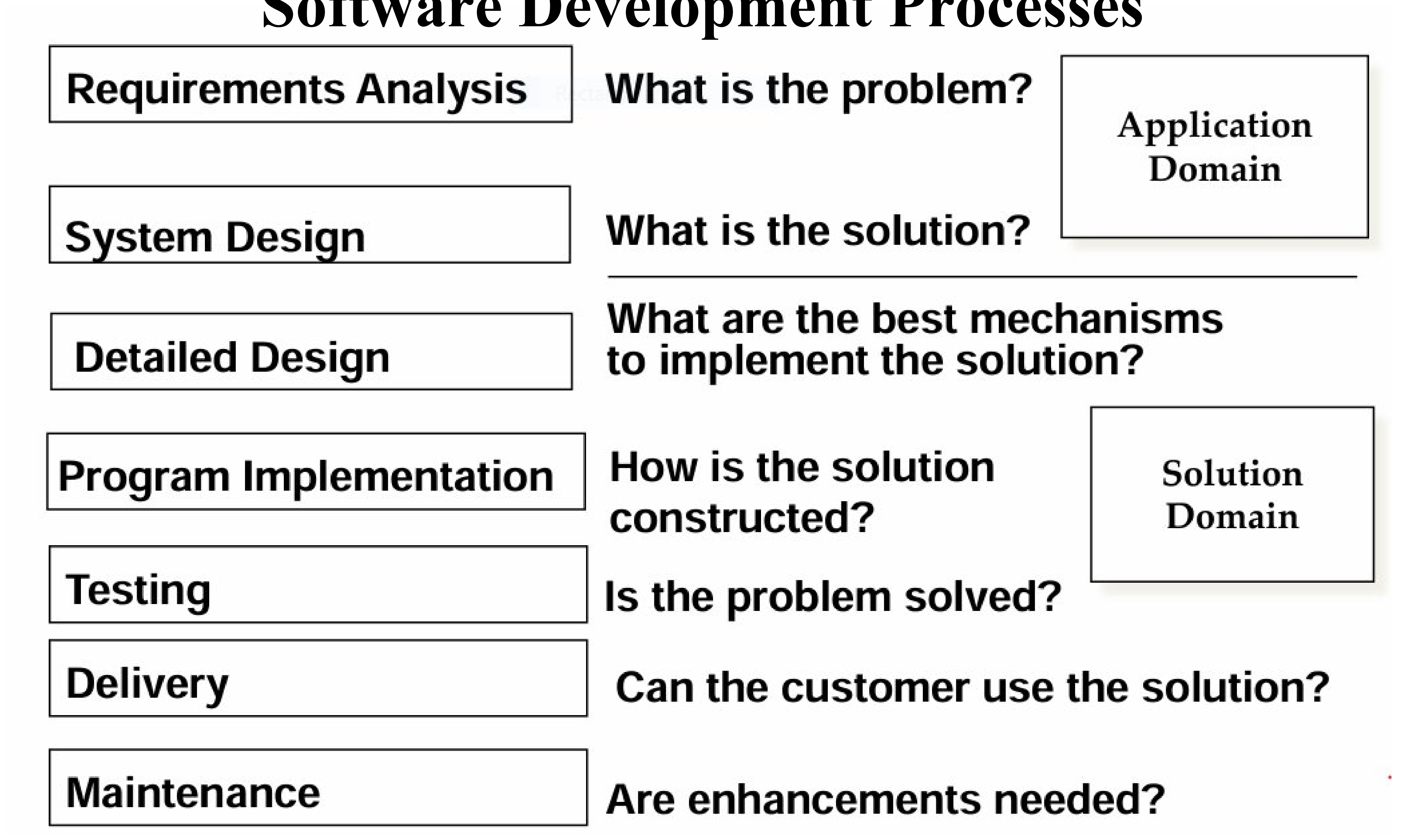
Chapter 02 – Software Process

Ms. Vithuckshiha Puvanasingam
Lecturer (Prob)
Department of ICT

Software Process is a structured set of activities required to develop a software system.

Software processes are intended to develop **quality software products efficiently** in terms of time and resource expenditure.

Software Development Processes

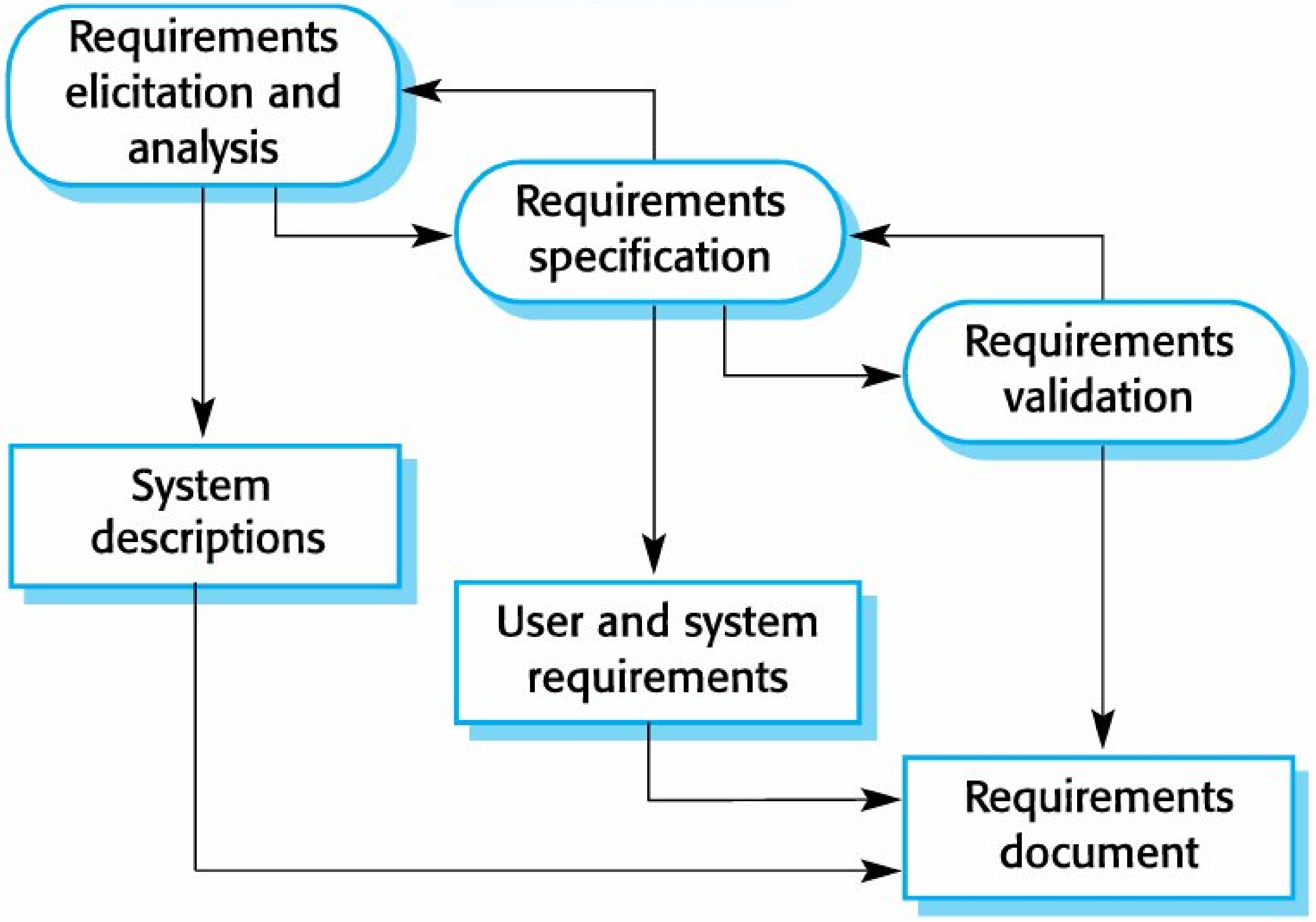


Requirements Analysis	What is the problem?	Application Domain
System Design	What is the solution?	
Detailed Design	What are the best mechanisms to implement the solution?	
Program Implementation	How is the solution constructed?	Solution Domain
Testing	Is the problem solved?	
Delivery	Can the customer use the solution?	
Maintenance	Are enhancements needed?	

Four fundamental activities are common to all software processes.

- **Software Specification**: defining what the system should do.
- **Software Development [Design and implementation]**: defining the organization of the system and implementing the system.
- **Software Validation**: checking that it does what the customer wants.
- **Software Evolution**: modified to reflect changing customer and market requirements.

The Requirement Engineering Process : Understanding and defining **what services are required from the system** and identifying the **constraints on the system's operation and development.**


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

Software Specification

Based on Requirement Engineering Process;

- **Requirements elicitation and analysis**

What do the system stakeholders require ?

- **Requirements Specification**

Defining the requirements in detail

- **Requirements Validation**

Checking the validity of the requirements

Software Development

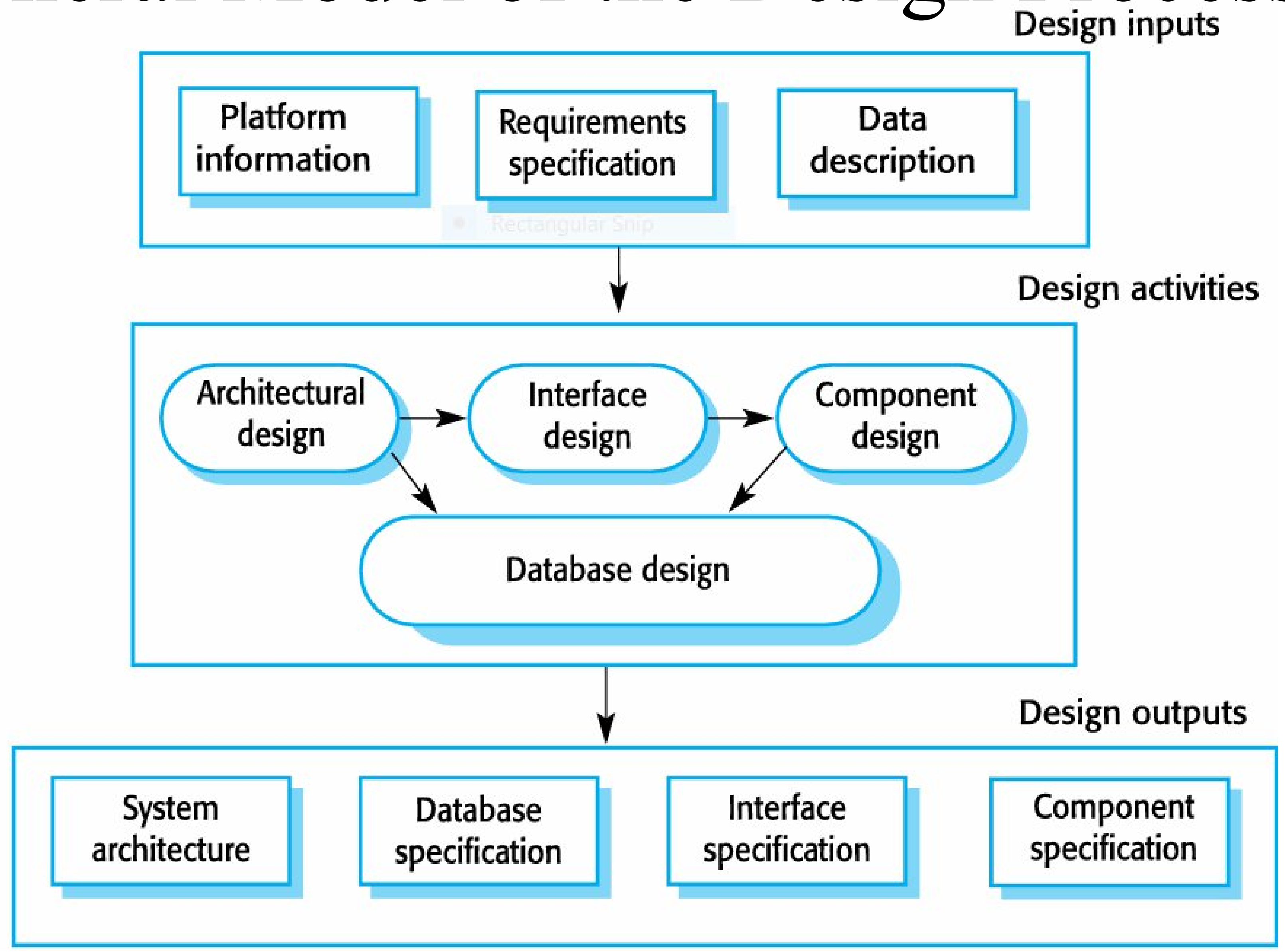
The process of **converting** the **system specification** into an **executable system**.

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 design, where you **take each system component and design** how it will operate and where you search for reusable components. If unavailable, you design how it will operate.

Design Phase	**Focus**	**Output**
Architectural	Structure, components, and their relationships	Architectural Specification(System blueprint)
Database	Data organization and storage	Database Specification(Database schema)
Interface	Communication between components	Interface Specification (API specification)
Component	Internal workings of modules	Component Specification (designs or code)

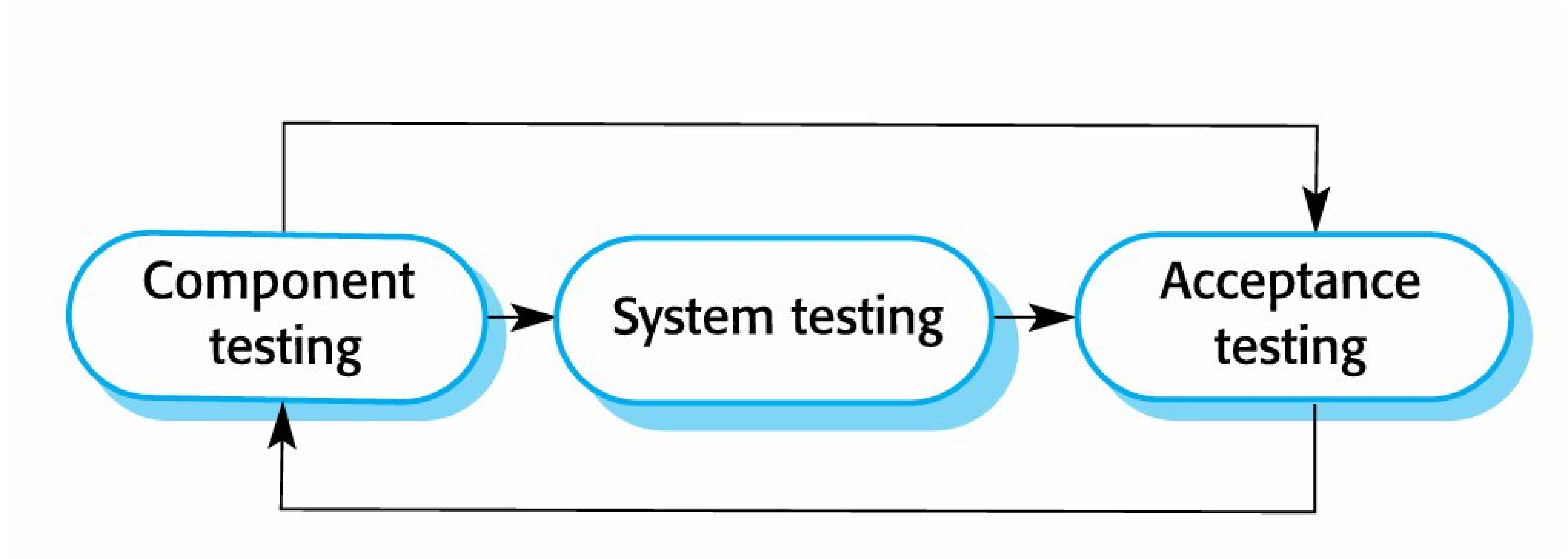
System Implementation

- 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 processes 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



- 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.

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

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 separation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.

Software Process Model

- A software process model represents the order in which the activities of software development will be undertaken. It describes the sequence in which the phases of the software life cycle will be performed.
- The **goal** of a software process model is to provide guidance for systematically coordinating and controlling the activities that must be performed in order to achieve the end product and the project objectives.

There are various **software development life cycle models** defined and designed which are followed during software development process. They are also called Software process Models:

Traditional Process Models:

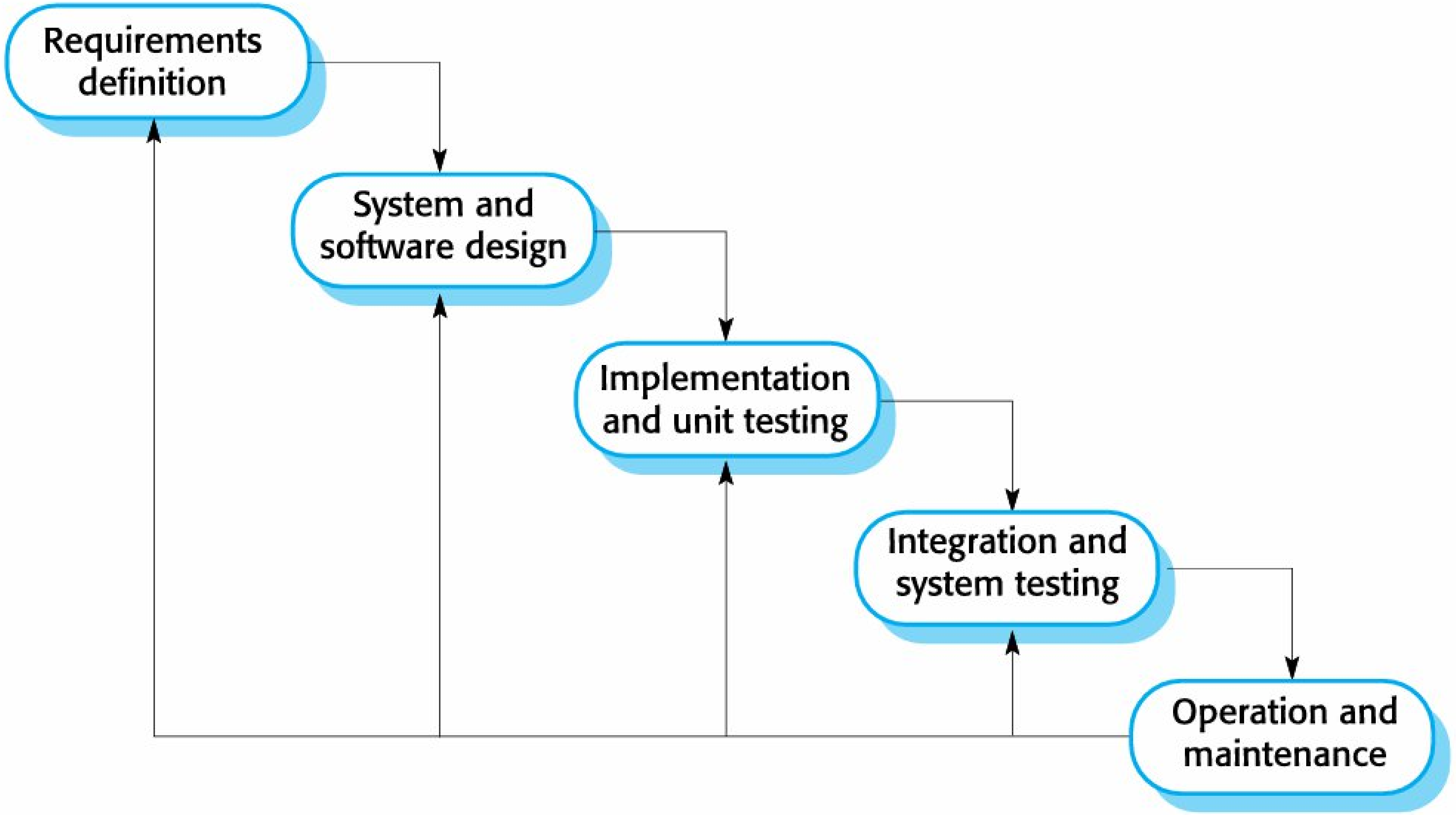
- **Waterfall**: a linear framework
- **Prototyping**: an iterative framework
- **Incremental**: a combined linear-iterative framework
- **Spiral**: a combined linear-iterative framework
- **Rapid application development (RAD)**: an iterative framework

Emerging Process Model :**Agile Model**- combination of iterative and incremental process models.

Model	**Type**	**Key Feature**
Waterfall	Linear	Step-by-step, no going back
Prototyping	Iterative	Build and refine with customer input
Incremental	Linear + Iterative	Add parts incrementally
Spiral	Linear + Iterative	Focus on risk assessment
RAD	Iterative	Fast development and feedback
Agile	Iterative + Incremental	Flexible, collaborative, adaptive

The Waterfall Model

- The Waterfall Model **was first Process Model** to be introduced. It is also referred to as a linear-sequential life cycle model or classic software life cycle or sequential model or traditional waterfall process model.
- It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.


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

When Waterfall Model

- Works for well understood problems with minimal or no changes in the requirements.
- Requirement is clear.
- Application is not complicated and big.
- The project is short.
- Product definition is stable. Technology and tools used are not dynamic and is stable.
- There are no ambiguous requirements. Sufficient resources with required expertise are available freely and trained.

Waterfall Model - Advantages

- Simple and easy to understand and use
- Organized approach with specific stages
- Reflects common engineering practices
- It is documentation driven, that is, documentation is produced at every stage.
- The output from every stage is fed into the next stage in sequence.
- Clearly defined stages.

Waterfall Model - Disadvantages

- Waterfall development is that it does not allow much reflection or revision.
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
- Development teams might wait for each other
- No working software is produced until late during the life cycle.
- All requirements must be known upfront

Incremental Development Model

- Incremental development model is based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate system has been developed.

When to Use

- Major requirements must be defined; however, some details can evolve with time.
- There is a need to get a product to the market early.
- This model is basically used when there is a limited time or the number of developers are less.
- Develop an Existing system

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.
- Parallel development can be planned.
- Testing and debugging during smaller iteration is easy.
- 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.

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 increasingly difficult and costly. software changes becomes
- More resources may be required.
- More management attention is required.

Prototyping

- Software prototyping, is the development approach of activities during software development, the creation of prototypes, i.e., incomplete versions of the software program being developed.
- Prototyping is the practice of **building an early version of a system** which does **not necessarily reflect all the features of the final system**, but rather those which are of interest.

- Small-scale mock-ups of the system are developed following an iterative modification process until the prototype evolves to meet the users’ requirements.
- In prototyping, the customer is presented at a very early stage with a working version of the system. (It may not be a complete system, but it is at least part of the system and it works.)
- The developer amends the system and demonstrates to the customer again and again until it does what the customer wants.

When Prototyping

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems

Advantages - Prototyping

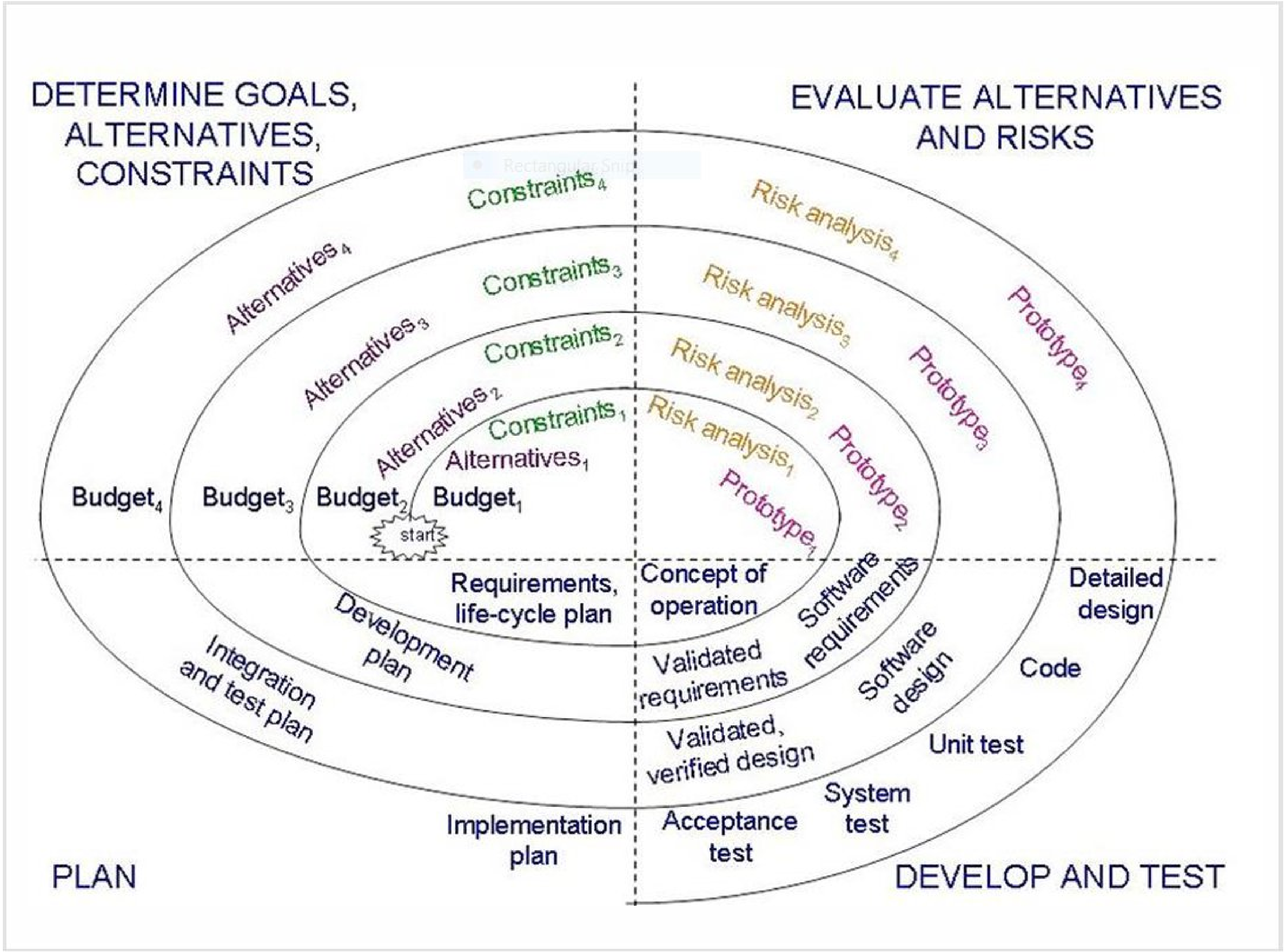
- Prototypes may be easily changed.
- Prototyping may improve communication between and among developers and customers.
- Users are actively involved in the development.
- Errors can be detected much earlier.
- Reduced time and costs
- Working SW available at early stages.

Disadvantages - Prototyping

- Prototyping may encourage an excess of change requests.
- Leads to implementing and then repairing way of building systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Incomplete application may cause application not to be used as the full system was designed Incomplete or inadequate problem analysis.
- Working prototypes may lead management and customers to believe that the final product is almost ready for delivery

Spiral Model

- The spiral model is similar to the incremental model, with more emphasis placed on **risk analysis**.
- Combines development activities with risk management to minimize and control risks.
- The model is presented as a **spiral** in which each iteration is represented by a circuit around four major activities– **Plan– Determine goals, alternatives, and constraints– Evaluate alternatives and risks– Develop and test**
- A software project repeatedly passes through these phases in iterations (called Spirals in this model)..


DETERMINE GOALS, ALTERNATIVES, CONSTRAINTS
EVALUATE ALTERNATIVES AND RISKS
Constraints$_4$
Alternatives$_4$
Constraints$_3$
Alternatives$_3$
Constraints$_2$
Alternatives$_2$
Constraints$_1$
Alternatives$_1$
Risk analysis$_4$
Risk analysis$_3$
Risk analysis$_2$
Risk analysis$_1$
Prototype$_4$
Prototype$_3$
Prototype$_2$
Prototype$_1$
Budget$_4$
Budget$_3$
Budget$_2$
Budget$_1$
start
Requirements, life-cycle plan
Concept of operation
Software requirements
Development plan
Validated requirements
Software design
Detailed design
Integration and test plan
Validated, verified design
Code
Unit test
System test
Acceptance test
Implementation plan
PLAN
DEVELOP AND TEST

When Spiral Model

- When costs and risk evaluation is important
- For medium to high-risk projects
- Users are unsure of their needs
- Requirements are complex
- New product line

Advantages – Spiral Model

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the software life cycle.

Disadvantages – Spiral Model

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project’s success is highly dependent on the risk analysis phase.
- Doesn’t work well for smaller projects.

RAD Model

- In Rapid Development Application model the **components or functions are developed in parallel** as if they were mini projects.
- This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

When RAD Model

- A system needs to be produced in a short span of time. (2-3 months)and The requirements are known.
- The user will be involved all through the life cycle.
- Technical risk is less.
- There is a necessity to create a system that can be modularized in 2-3 months of time.
- A budget is high enough to afford designers for modeling along with the cost of automated tools for code generation.

Advantages - RAD

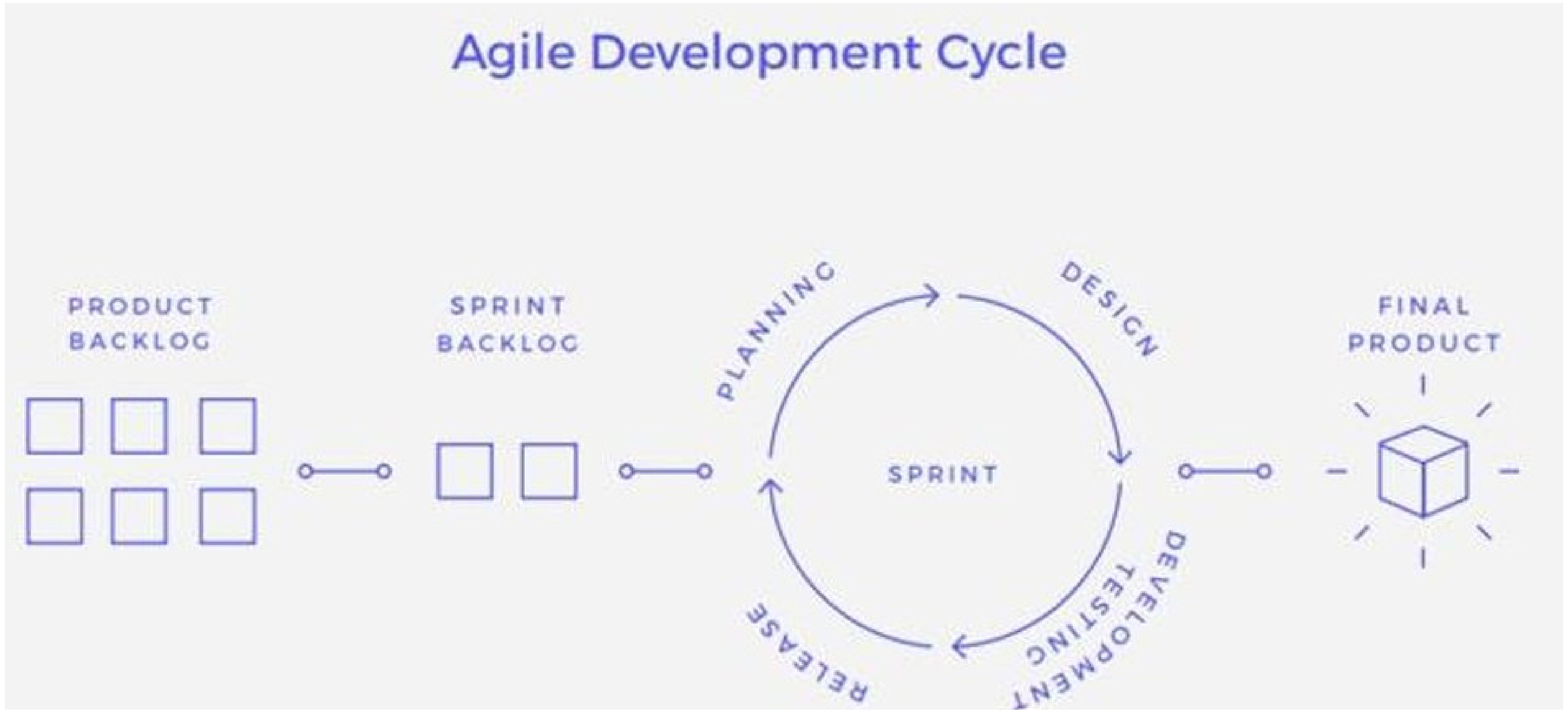
- Flexible and adaptable to changes.
- It is useful when you have to reduce the overall project risk.
- Due to code generators and code reuse, there is a reduction of manual coding.
- Each phase in RAD delivers highest priority functionality to client.
- With less people, productivity can be increased in short time.

Disadvantages - RAD

- It can’t be used for smaller projects. Only systems that can be modularized can be built using RAD.
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.
- When technical risk is high, it is not suitable.
- Requires highly skilled developers/ designers. High dependency on modeling skills.

Agile Model

- Agile Process Model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.
- Agile Methods break the product into small incremental builds. These builds are provided in quick time iterations. Every iteration involves cross functional teams working simultaneously on basic activities like requirement analysis and design .
- At the end of the iteration, a working product is displayed to the customer.
- Agile uses an adaptive approach where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed.



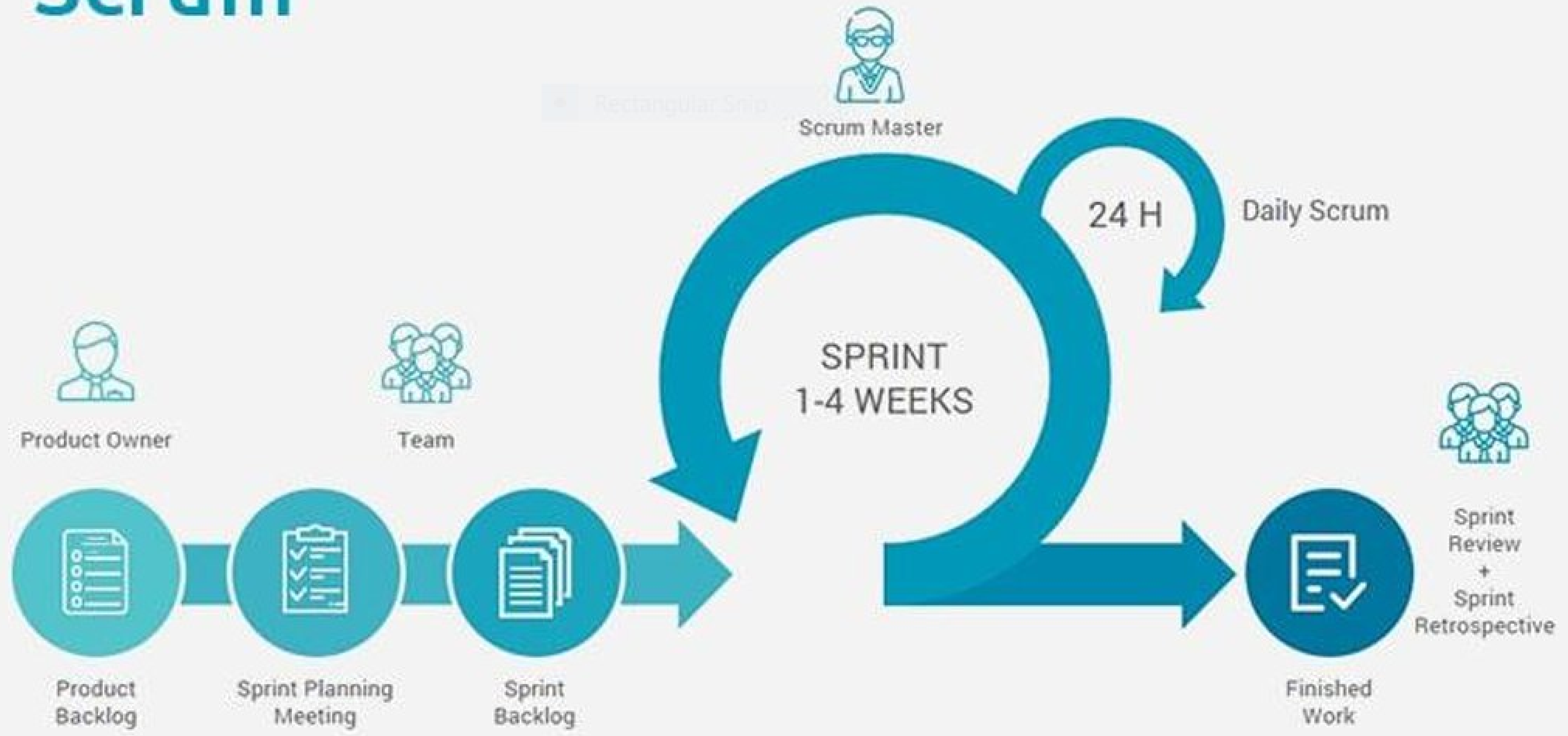
Sprint is one time boxed iteration of a continuous development cycle. Within a Sprint, planned amount of work has to be completed by the team and made ready for review.

The agile **product backlog** is a prioritized features list, containing short descriptions of all functionality desired in the product.

How Agile Works

- Simply, consider each sprint to be a miniature project having its own backlog, design, development, testing, and deployment phases within the pre-defined scope of work.
- At the end of each sprint, a potentially deliverable product is shipped. In simple terms, with every iteration completion, new features are added to the main software which results into the software growth.
- The most popular Agile methods include **Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995)**. These are now collectively referred to as Agile Methodologies, after the **Agile Manifesto** was published in 2001.

Scrum


Scrum Master
24 H
Daily Scrum
Product Owner
Team
SPRINT
1-4 WEEKS
Product Backlog
Sprint Planning Meeting
Sprint Backlog
Finished Work
Sprint Review + Sprint Retrospective

Scrum Steps

- Step 1: Product Backlog Creation
- Step 2: Sprint Planning
- Step 3: Sprint Backlog Creation
- Step 4: Sprint
- Step 5: Output: Working Product
- Step 6: Sprint Review
- Repeat these work flow for each Sprint.

Advantages - Agile

- Working through Pair programming produce well written compact programs which has fewer errors as compared to programmers working alone.
- It reduces total development time of the whole project.
- Customer representative get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.

Disadvantages - Agile

- Due to lack of formal documents, it creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.
- Due to absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.