Processes

Object Orientated Analysis and Design

Benjamin Kenwright

Outline

- Review

- What are Processes?

- Why are they important in Object Orientated Analysis and Design

- Conclusion and Discussion

- Summary

Revision Question

An object is an entity that has:

a) state, action, dependencies

b) behavior, state, identity

c) behaviour, action, state

d) class, state, memory

Answer

b) behavior, state, identity

Revision Question

🟥Write down the four elements that Inherent Complexity derives from?

(5 Minutes)

Answer

■ Inherent Complexity derives from four elements:

1. complexity of the problem domain

2. difficulty of managing the development process

3. flexibility possible through software

4. problems of characterizing the behavior of discrete systems

Revision Question

- Which of the following is the functionality of 'Data Abstraction'?

- A. Reduce Complexity

- B. Binds together code and data
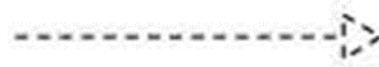
- C. Parallelism

- D. None of the mentioned

Answer

- A. Reduce Complexity

Explanation: An essential element of Object Oriented Programming is 'Data Abstraction' which means hiding things. Complexity is managed through abstraction.

Revision Question

- Draw the notations for the different types of relationships

Dependency   ------------- ▷

Association

Direct Association

Inheritance

Realization

Aggregation

Answer

Dependency	----------------▷▷
Association	——————————
Direct Association	—————————————▶
Inheritance	—————————————▷
Realization	--------------▶
Aggregation	——————————◇

Revision Question

■ List the various UML Diagram Types

Answer

Structure Diagrams	Behavior Diagrams
Package Diagram	Use Case Diagram
Class Diagram	Activity Diagram
Component Diagram	State Machine Diagram
Deployment Diagram	**Interaction Diagrams**
Object Diagram	Sequence Diagram / Communication Diagram
Composite Structure Diagram	Interaction Overview Diagram / Timing Diagram
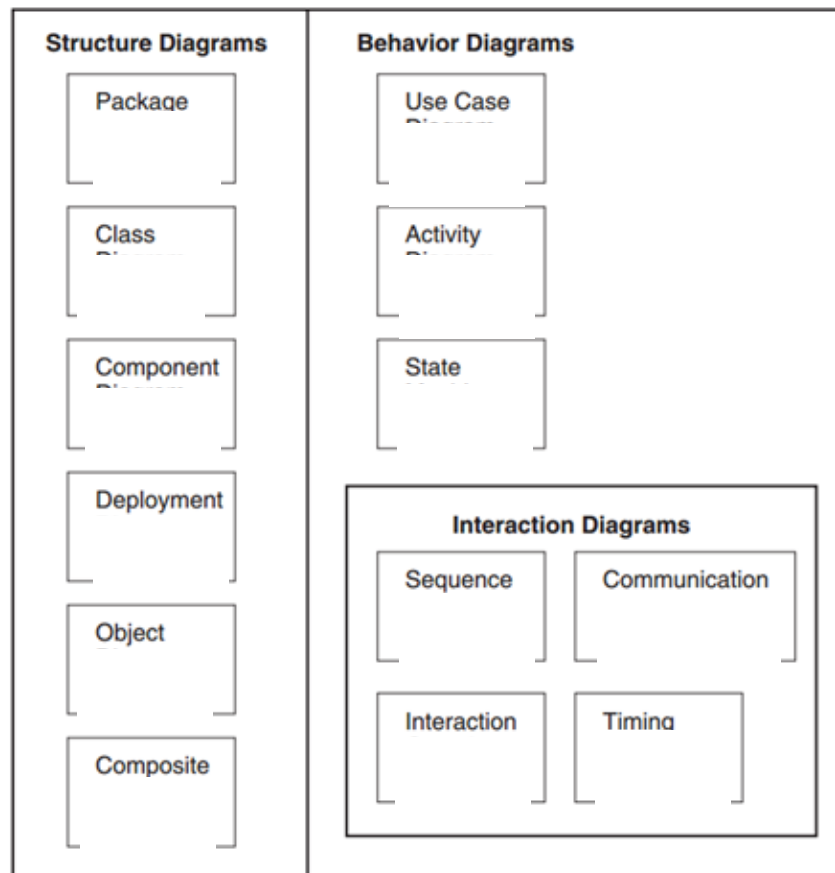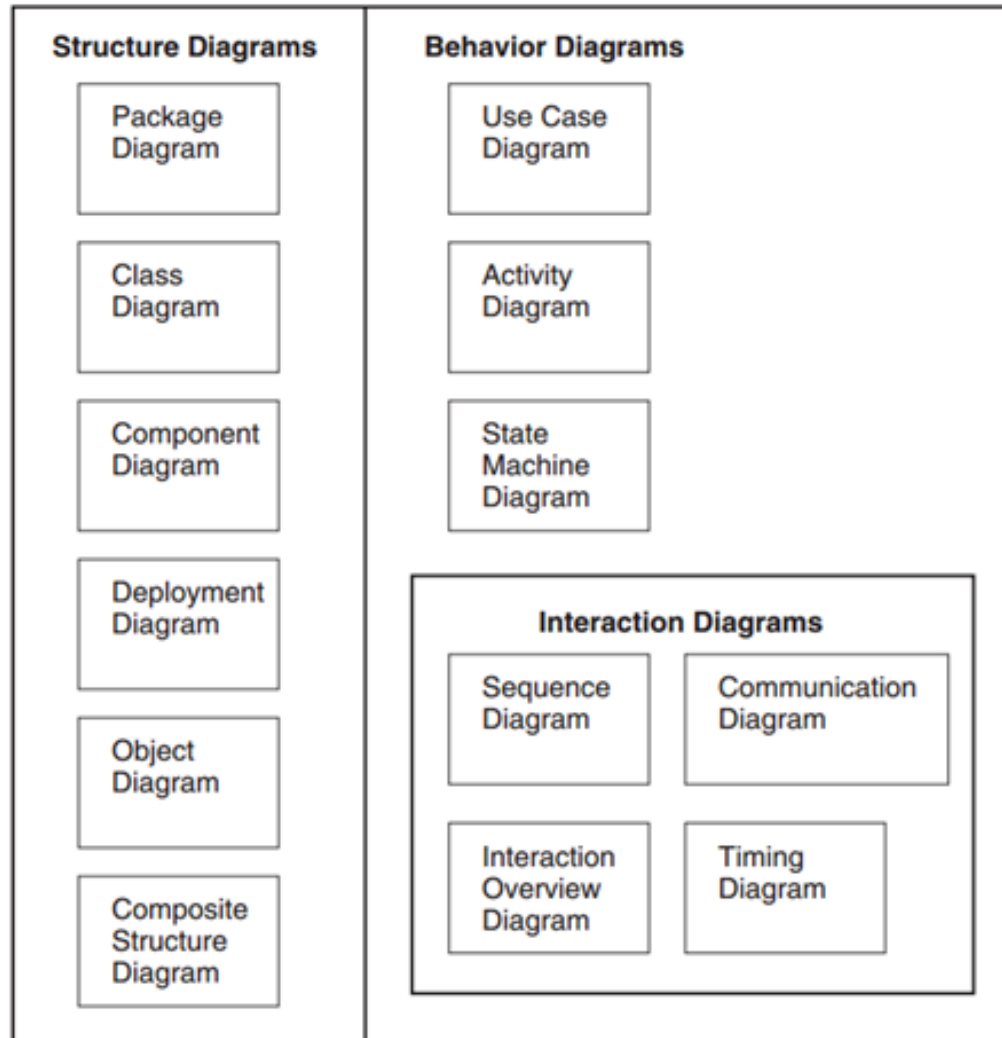
Question

■ Draw an example of a State Machine Diagram
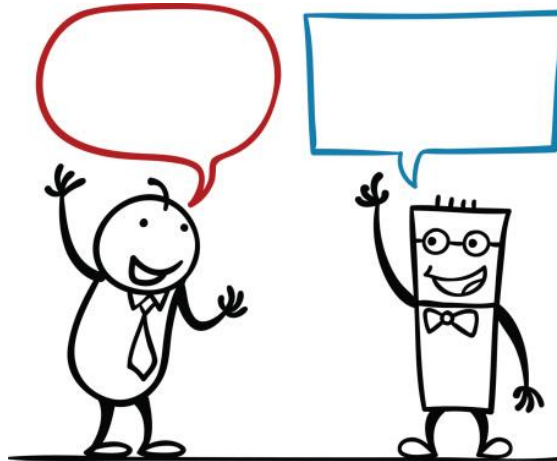
Answer

- A state machine diagram expresses behavior as a progression through a series of states, triggered by events, and the related actions that may occur
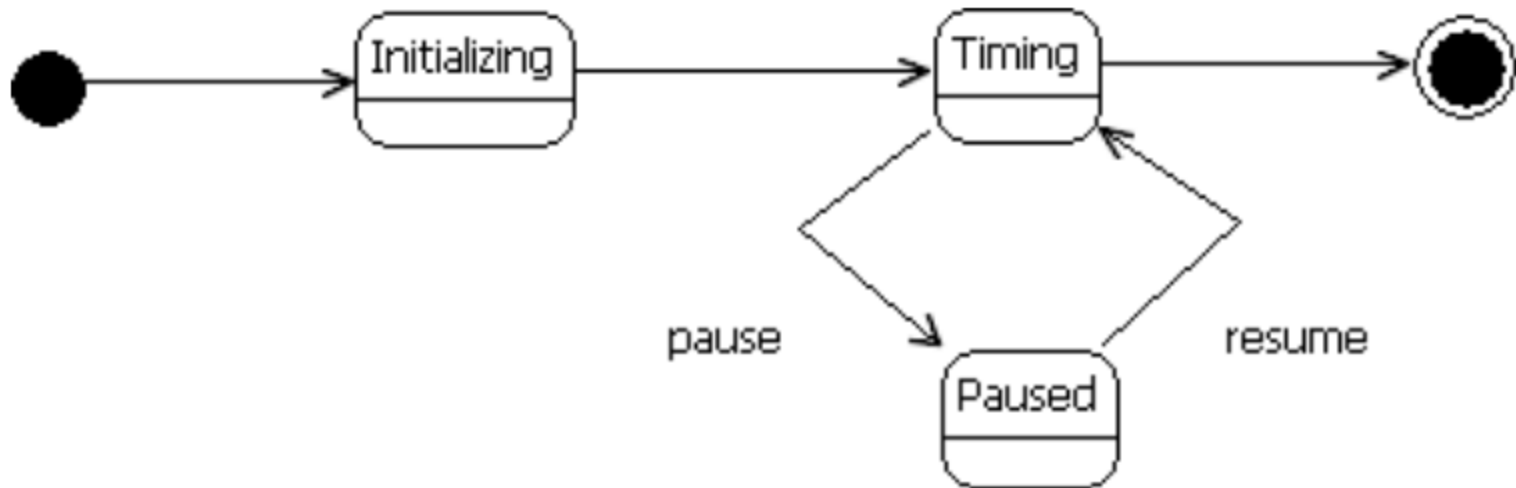
States and Transition Events for the Duration Timer

Question

- Does a method or tool exist which promises to make software development a trivial task?

a) True
b) False

Answer

■b) False

A professional software engineer know that no such *panacea* exists.

Traits of Successful Projects

- Existence of a strong architectural vision

- Application of a well-managed iterative and incremental development lifecycle

Note there are other traits of a successful project – however, we focus on just two main principles

Strong Architectural Vision

- Fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution

- Conceptual integrity

- Clean internal structure
  - ▷ Understandability

Attributes of Good Software Architectures

- Well-defined layers of abstraction
  - ▷ Each layer representing a coherent abstraction
  - ▷ Provides well-defined and controlled interface
  - ▷ Built on well-defined and controlled facilities at lower levels of abstraction
- Clear separation of concerns between the interface and implementation of each layer
  - ▷ Makes it possible to change the implementation of a layer without violating the assumptions made by its clients
- Architecture is simple
  - ▷ Common behavior is achieved through common abstractions and common mechanisms

Iterative and Incremental Lifecycle
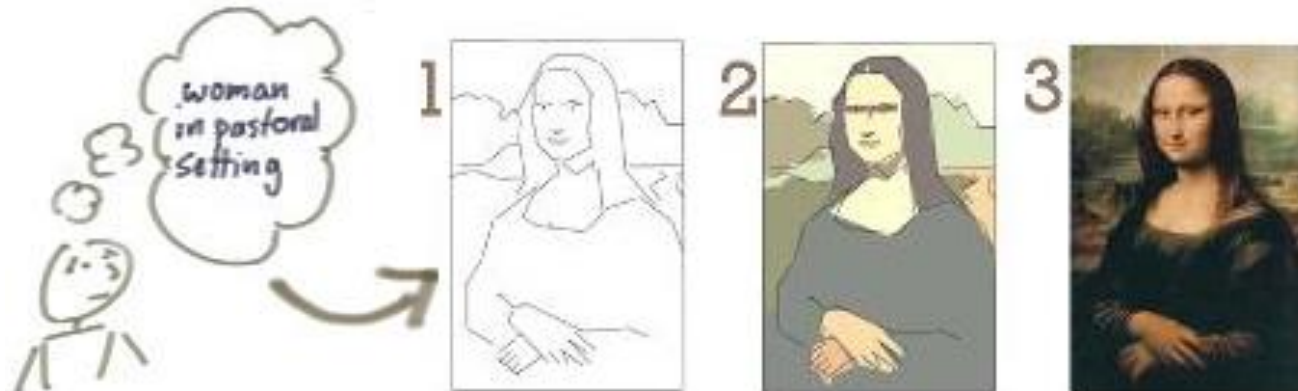
Incremental Approach

- The **Incremental Approach** uses a set number of steps and development goes from start to finish in a linear path of progression.

Incremental

- Incremental development is done in steps from design, implementation, testing/verification, maintenance. These can be broken down further into sub-steps but most incremental models follow that same pattern. The **Waterfall Model** is a traditional incremental development approach.

Iterative Approach

- **Iterative Approach** has no set number of steps, rather development is done in cycles

Iterative

- Iterative development is less concerned with tracking the progress of individual features. Instead, focus is put on creating a working prototype first and adding features in development cycles where the Increment Development steps are done for every cycle. **Agile Modeling** is a typical iterative approach.

Analogy of Incremental vs Iterative

- **If you were writing an essay under the Incremental Model, you'd attempt to write it perfectly from start to finish one sentence at at time. If you wrote it under the Iterative Model, you'd bang out a quick rough draft and work to improve it through a set of revision phases**

Question

- Which development approach is the the waterfall model?

a) incremental development approach

b) iterative development approach

c) static development approach

d) behavioral development approach

Answer

- a) incremental development approach

# Heart of Software Engineering

- Iterative and incremental approach are at the heart of most modern software development methods, including agile methods like:
  - ▷ Extreme Programming (XP)
  - ▷ SCRUM
- Most importantly are extremely well suited to the object-oriented paradigm and offers a number of benefits relative to risk management

Trends

- Iterative approach is common practice because it better fits the natural path of progression in software development
- Instead of investing a lot of time/effort chasing the 'perfect design' based on assumptions, the iterative approach is all about creating something that's 'good enough' to start and evolving it to fit the user's needs

Discussion

🟥When would you use an Iterative or Incremental approach?
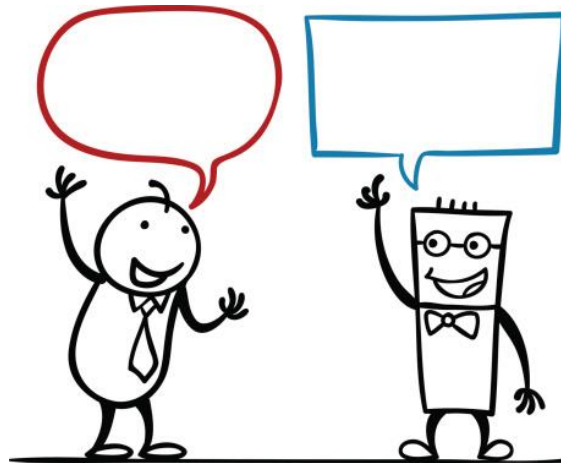
Plan-Driven vs Agile
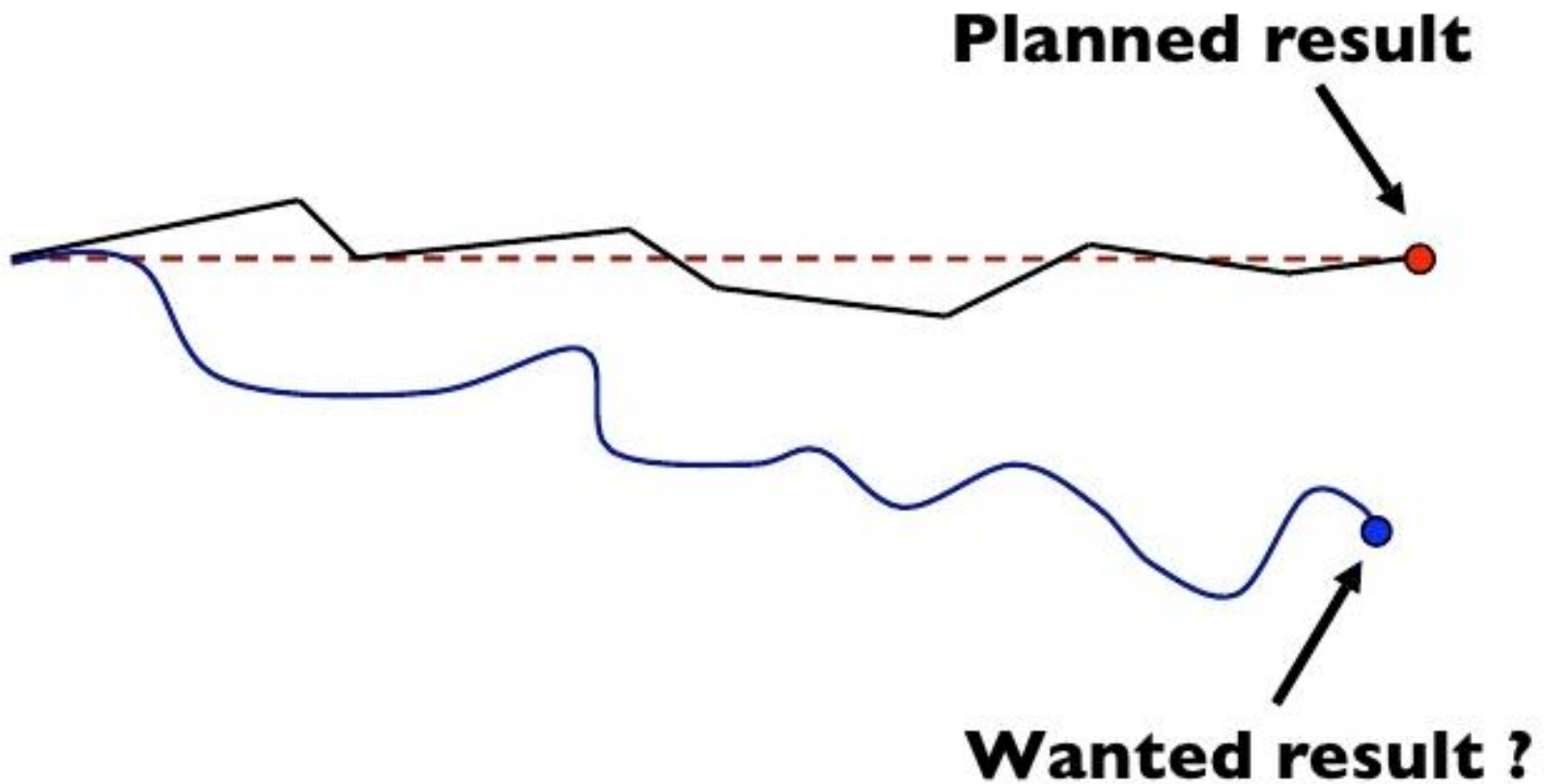
Agile
Tight Cycles and Small Increment

Waterfall (Plan-Driven)

Incremental

Agile-XP
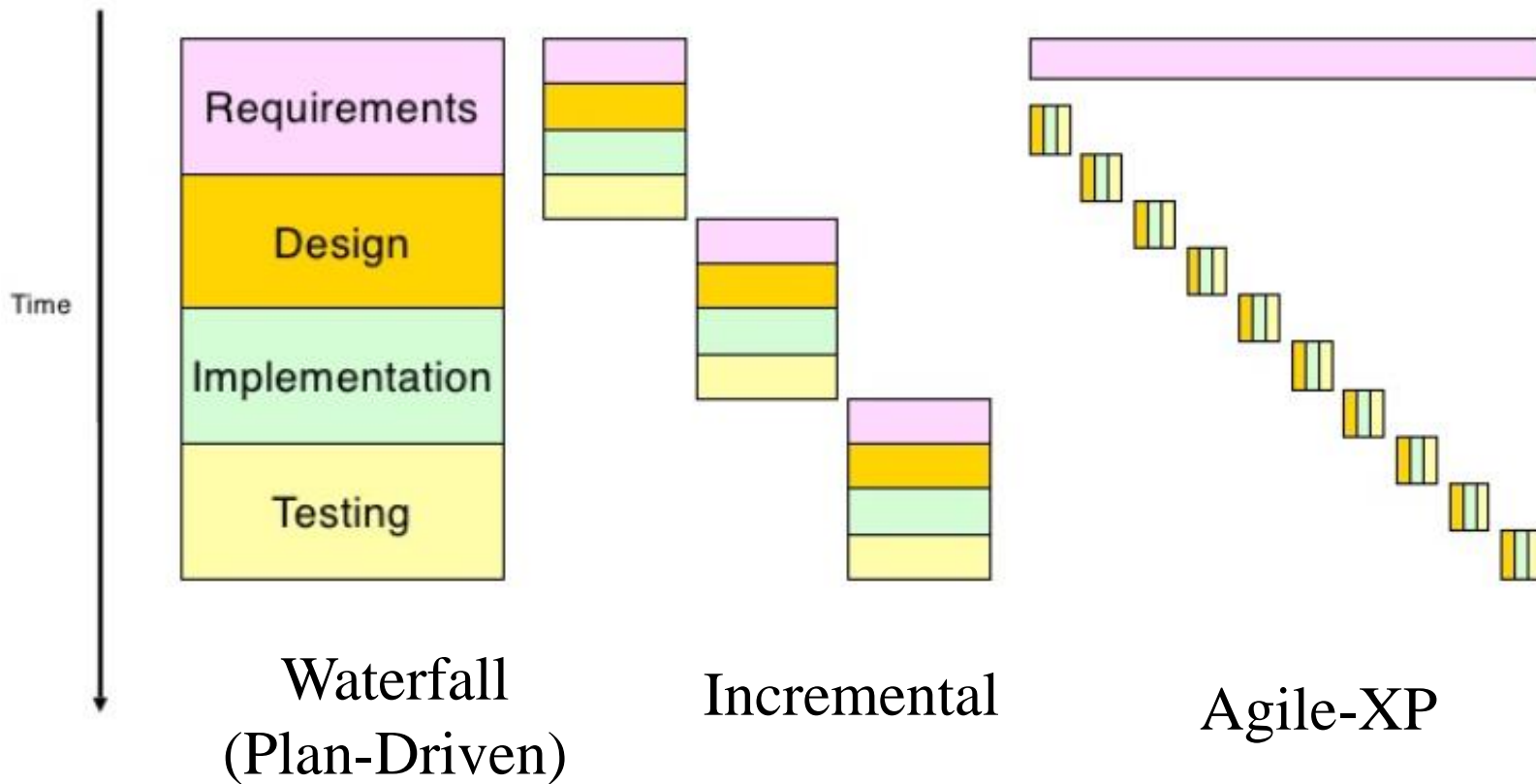
Agile Processes

- Primary goal is to deliver a system to the customer that meets their current needs in the shortest amount of time

Agile Characteristics

- Lightweight and sparse, less ceremony
- Reliant on the tacit knowledge of the team members
- Tactically focused rather than strategic
- Iterative and incremental
- Heavily reliant on customer collaboration
- Self-organizing and managing
- Emergent as opposed to predetermined

Plan-Driven Characteristics

- More heavyweight, more ceremony
- Reliant on well-documented processes
- Strategically focused rather than tactically focused
- Reliant on a customer contract
- Managed and controlled
- Defined up front and then continually improved

Agile	Plan-Driven
■ Project is small	■ Project is large
■ Experienced teams with a wide range of abilities take part	■ Teams include varied capabilities and skill sets
■ Teams are self-starters, independent leaders and others who are self-directing	■ Teams are geographically distributed and/or outsourced
■ Project is an in-house project and the team co-located	■ Project is of strategic importance
■ System is new with lots of unknowns	■ System is well understood (scope and features set)
■ Requirements must be discovered	■ Requirements are fairly stable
■ Requirements and environment are volatile with high change rates	■ System is large and complex (critical safety/high reliability requirements)
■ End-user environment is flexible	■ Project stakeholders have a weak relationship with the development team
■ Relationship with customer is close and collaborative	■ External legal concerns
■ Customer is readily available dedicated and co-located	■ Focus is on a strong, quantitative process improvement
■ High trust environment exists within the development teams and customer	■ Definition and management of process are important
■ Rapid value and high-responsiveness are required	■ Predictability and stability of process are important

Software Development Lifecycle (SDLC)
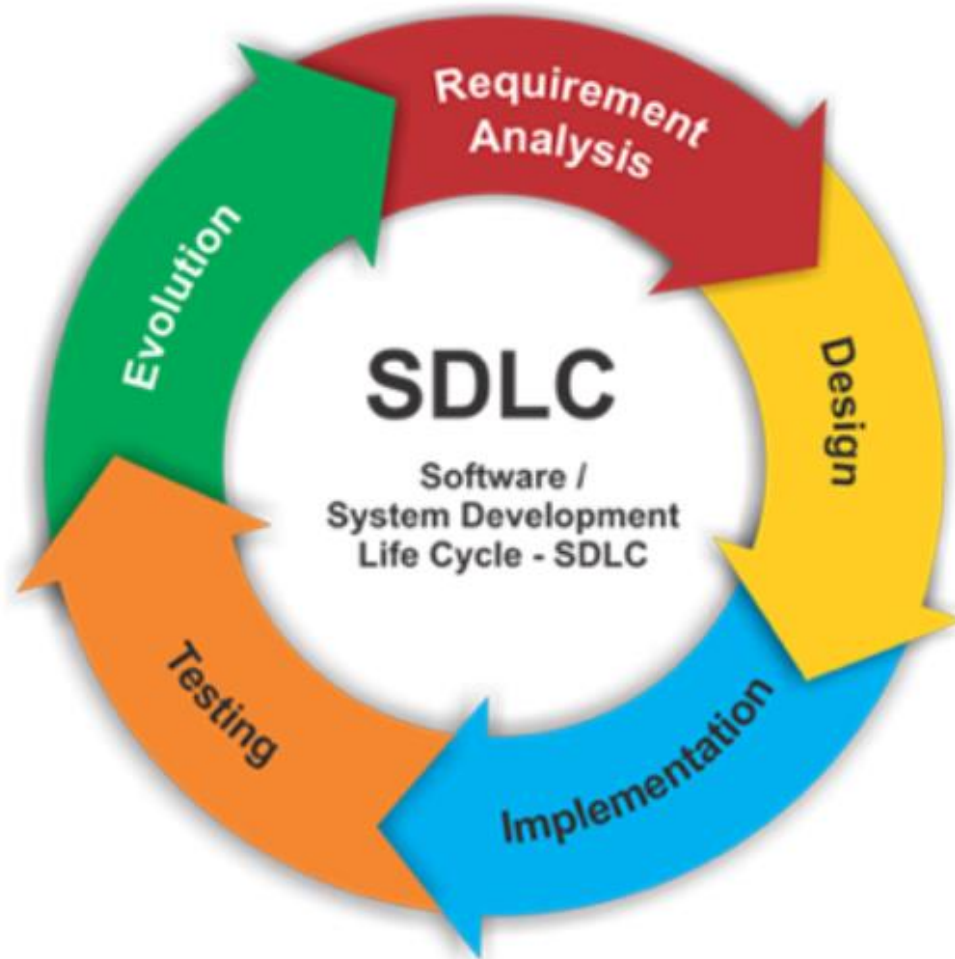
Software Development Lifecycle

■ Controlling framework

■ Guide for the overall development of the system

▷ ultimately leading to the final product

Selection

- Plethora of software development lifecycle styles available

  ▷ For example, Rational Unified Process (RUP), XP, SCRUM, Crystal, ...

- Selection of a lifecycle style directly affects the size and shape of the process

Five Process Areas

1. Requirements
2. Analysis and Design
3. Implementation
4. Testing
5. Deployment

| Requirements | → | Analysis & Design | → | Implementation | → | Test | → | Deployment |
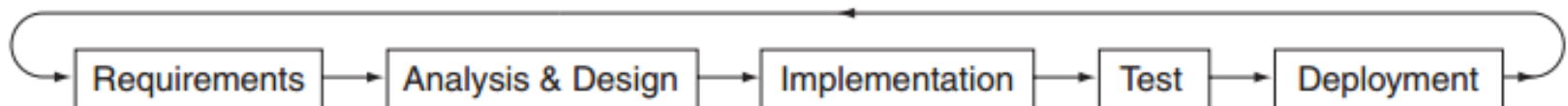
Throughout the Lifecycle

- *Project Management*
  - ▷ Manage the software development project, including:
    - planning, staffing, and monitoring the project, as well as managing the risks
- Configuration and Change Management
  - ▷ Identify and control change
- Environment
  - ▷ Software development environment, including teams, tools, and support

Process Disciplines

Requirements	Analysis & Design	Implementation	Test	Deployment

Project Management
Configuration and Change Management
Environment
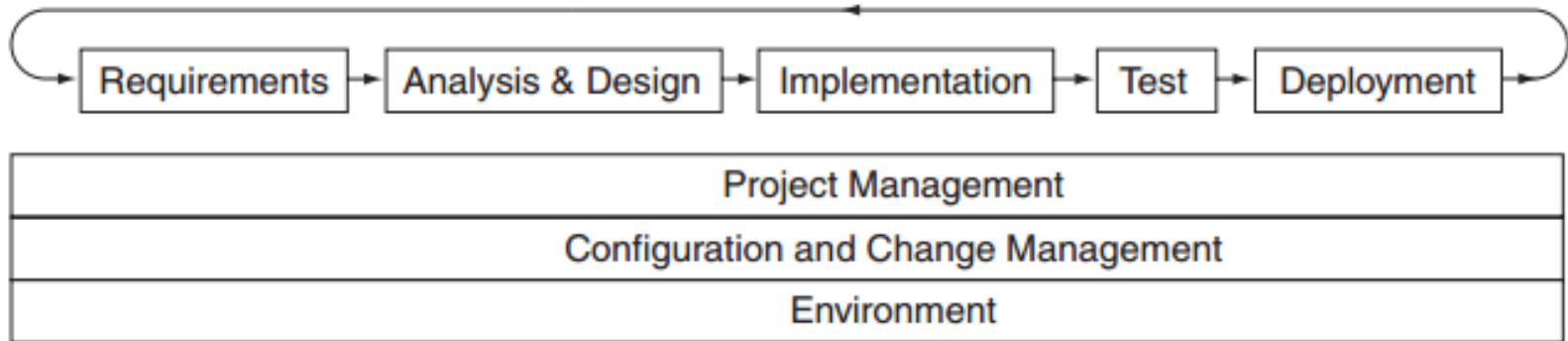
◼ Relative Order

◼ Iterative Nature

Process Milestones, Phases and Iterations
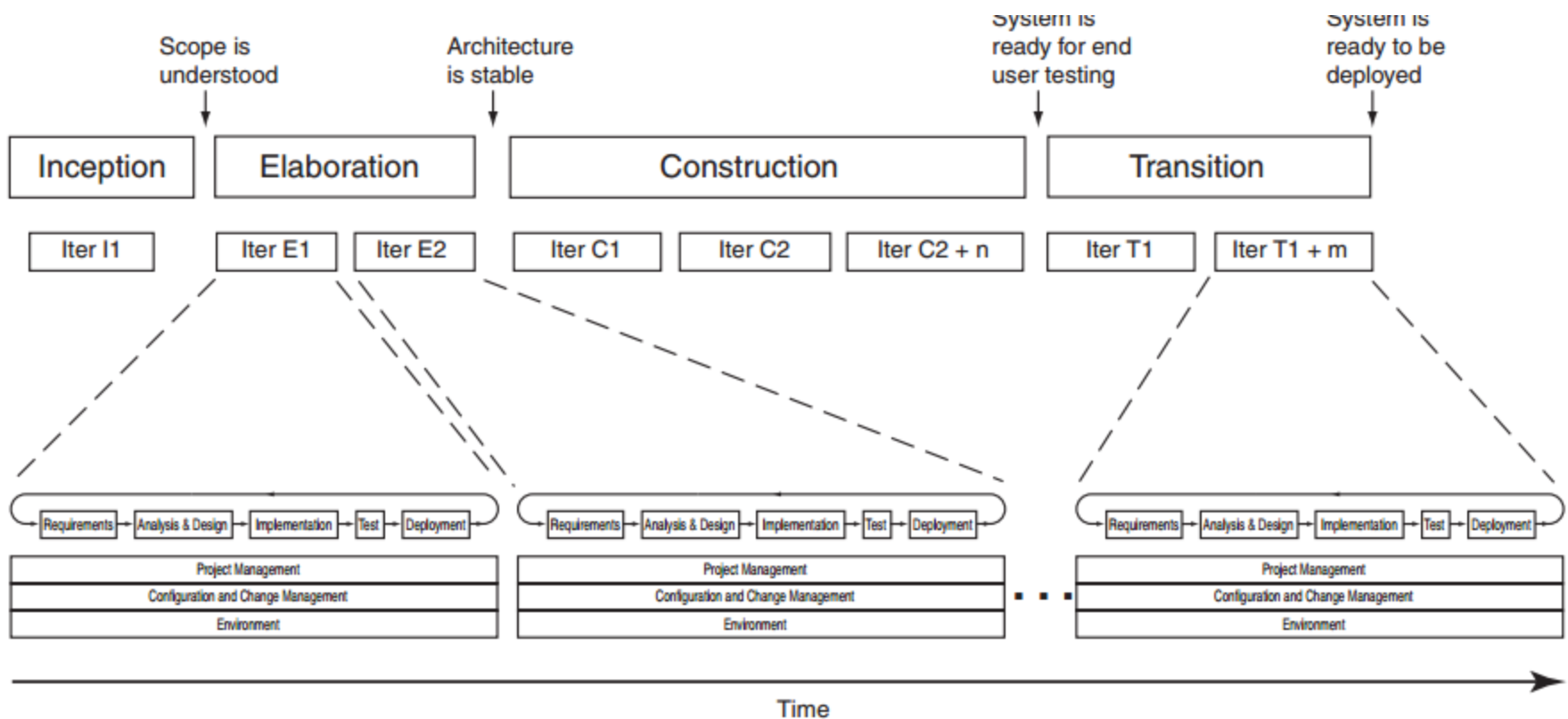
Phases in Agile Methods

■ The XP lifecycle includes five phases

1. *Exploration*: Determine feasibility, understand key "stories" for the first release, and develop exploratory prototypes
2. *Planning*: Agree on the date and stories for the first release
3. *Iterations to release*: Implement and test selected stories in a series of iterations. Refine the iteration plan
4. *Productionizing*: Prepare supporting materials (documentation, training, marketing), and deploy the operational system
5. *Maintenance*: Fix and enhance the deployed system

Phases in Agile Methods

■ SCRUM lifecycle includes four phases.

1. *Planning*: Establish the vision, set expectations, secure funding, and develop exploratory prototypes

2. *Staging*: Prioritize and plan for the first iteration. Develop exploratory prototypes

3. *Development*: Implement requirements in a series of sprints, and refine the iteration plan

4. *Release*: Prepare supporting materials (documentation, training, marketing), and deploy the operational system

Duration

- Iteration is pretty much the same across most software development methods

- XP recommends that iterations be one or two weeks long, if possible
- SCRUM specifies that all iterations (sprints) should be 30 days long
- RUP recommends that iterations be two to six weeks long

Micro Process

- Overall software development process (the macro process)

- Micro Process covers the analysis and design process by looking at what activities are performed and what work products are produced
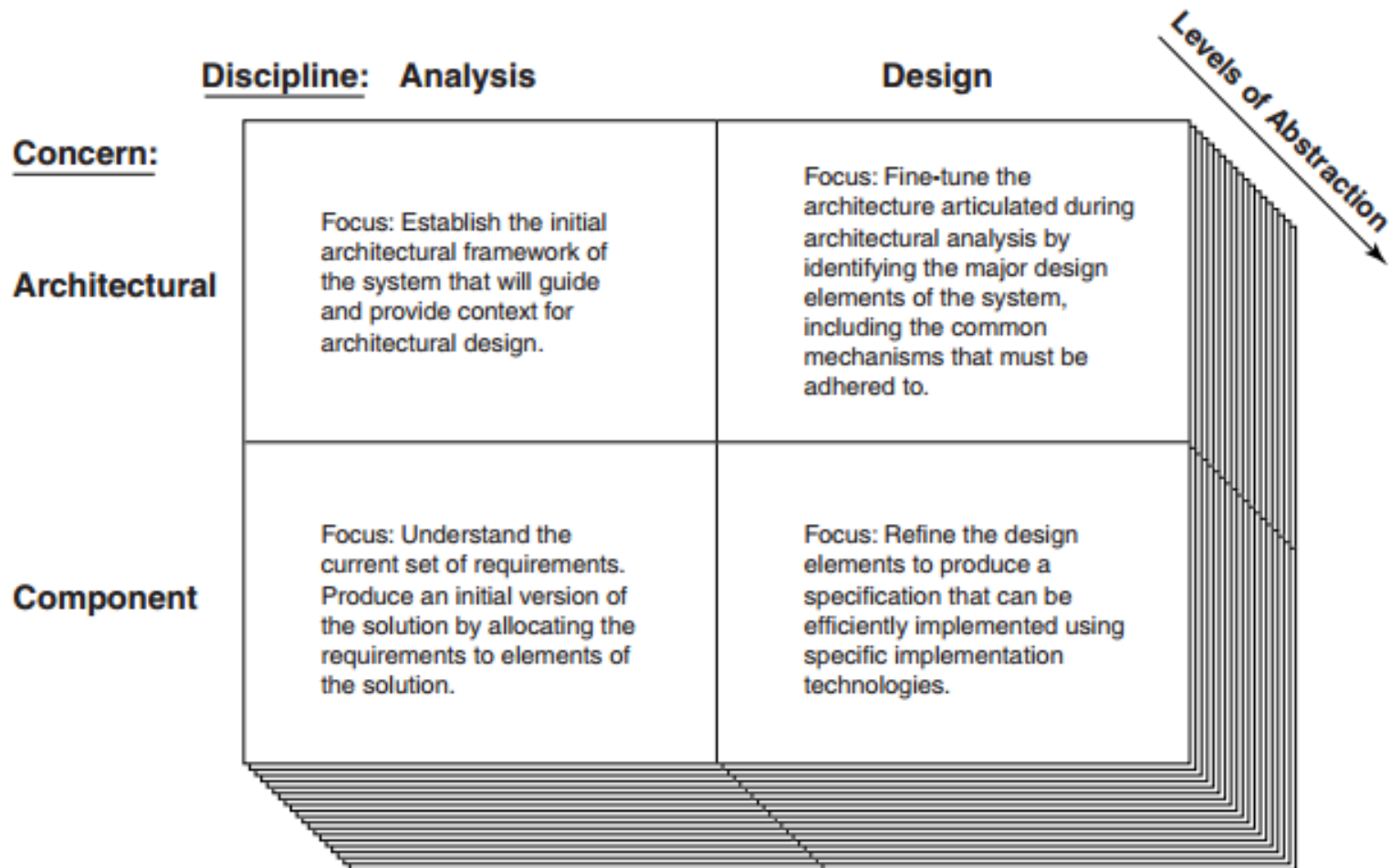
# Micro Process within Macro Process
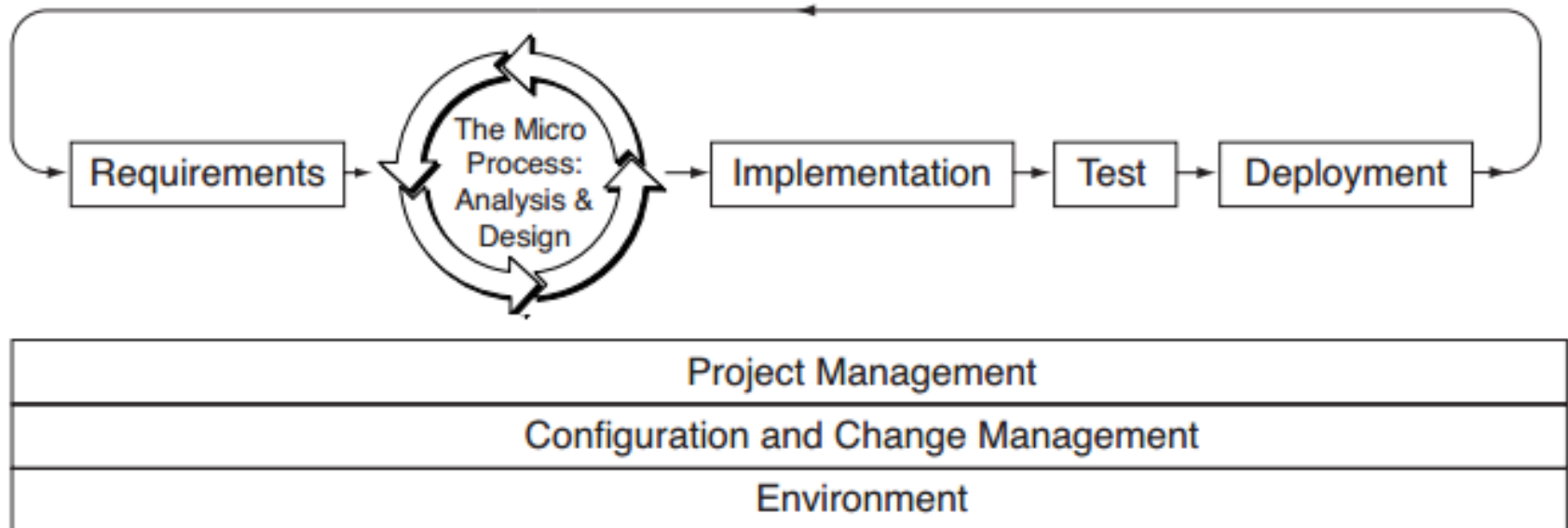
Levels of Abstraction

- Micro process, the traditional phases of analysis and design are intentionally blurred and instead are performed at different levels of abstraction along a continuum

Vary Focus of Analysis and Design Depending on Perspective

Discipline: Analysis   **Design**   *Levels of Abstraction*

Concern:

Architectural

Focus: Establish the initial architectural framework of the system that will guide and provide context for architectural design.

Focus: Fine-tune the architecture articulated during architectural analysis by identifying the major design elements of the system, including the common mechanisms that must be adhered to.

Component

Focus: Understand the current set of requirements. Produce an initial version of the solution by allocating the requirements to elements of the solution.

Focus: Refine the design elements to produce a specification that can be efficiently implemented using specific implementation technologies.

Activities

- Identify the elements
- Define the collaborations between the elements
- Define the relationship between elements
- Define the semantics of the elements
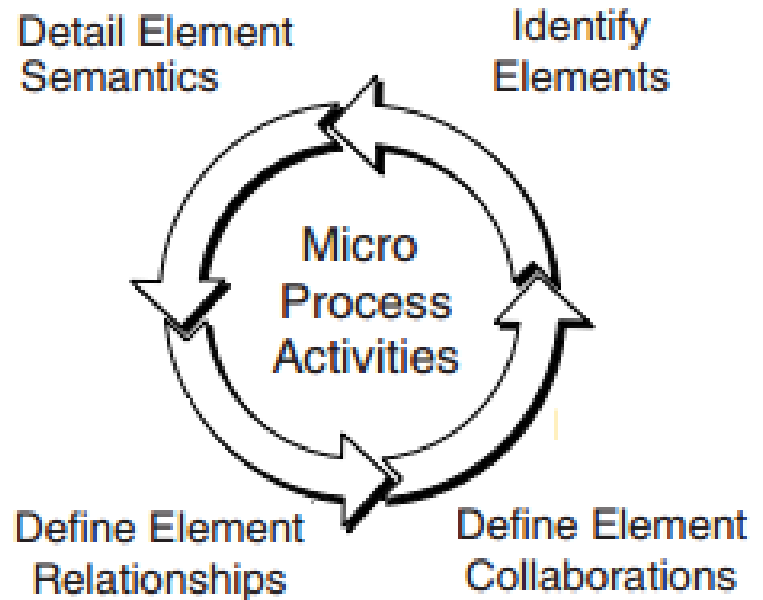
Review

- **Review Six Common Software Development Lifecycles (SDLC)**
  - ▷ Waterfall Model
  - ▷ V-Shaped Model
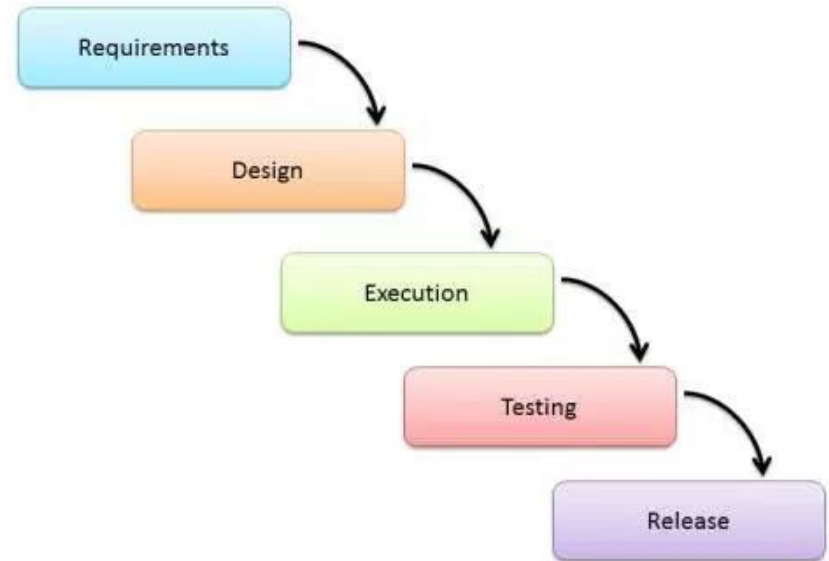 - ▷ Evolutionary Prototyping Model
  - ▷ Spiral Method (SDM)
  - ▷ Iterative and Incremental Method
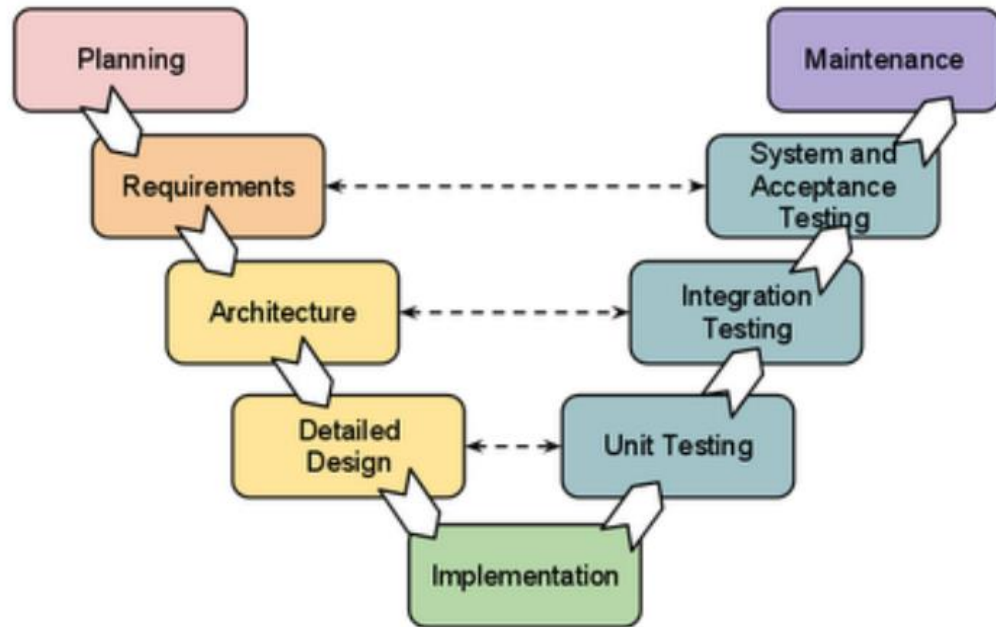 - ▷ Agile development

Waterfall Model

- Linear sequential flow
- Projects which do not have changing requirements

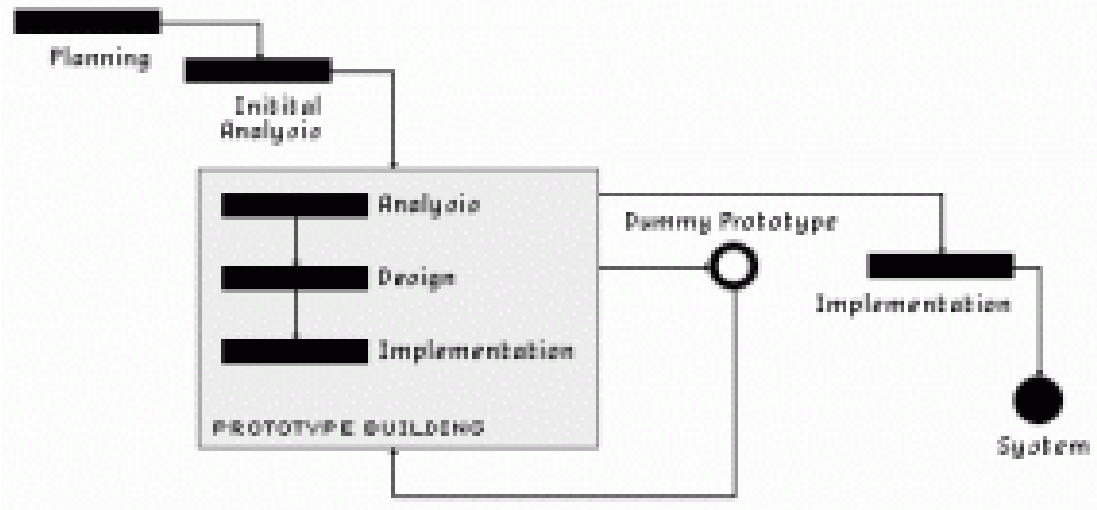| Advantages	Disadvantages
- Easy to explain to the users. - Structures approach. - Stages and activities are well defined. - Helps to plan and schedule the project. - Verification at each stage ensures early detection of errors/misunderstanding. - Each phase has specific deliverables.	- Assumes that the requirements of a system can be frozen. - Very difficult to go back to any stage after it finished. - A little flexibility and adjusting scope is difficult and expensive. - Costly and required more time, in addition to the detailed plan.

V-Shaped Model

■ Difference between V-shaped model and waterfall model is the early test planning in the V-shaped model

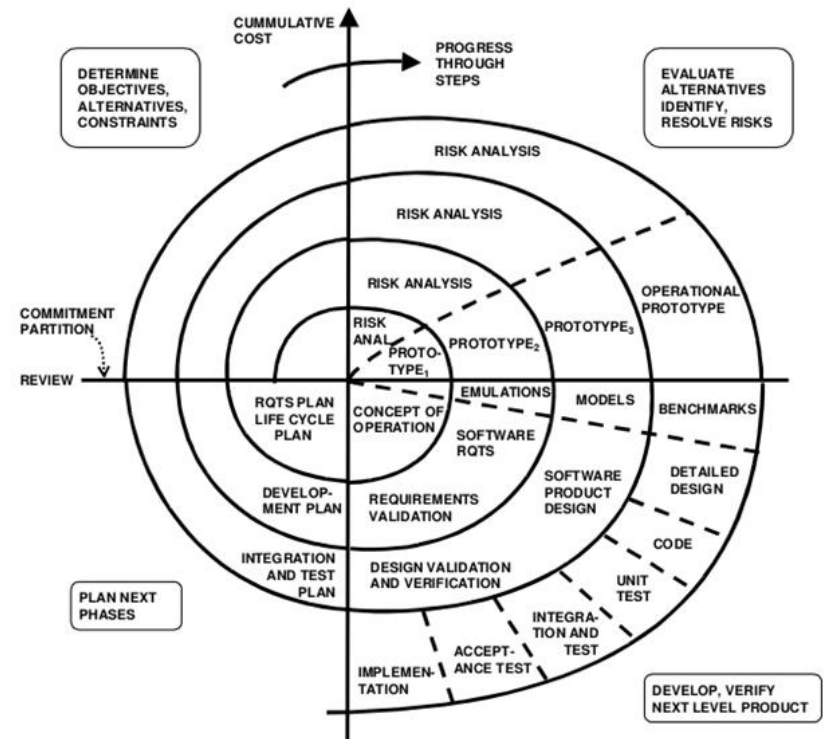| Advantages	Disadvantages
■ Simple and easy to use ■ Each phase has specific deliverables. ■ Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle. ■ Works well for where requirements are easily understood. ■ Verification and validation of the product in early stages of product development.	■ Very inflexible, like the waterfall model. ■ Adjusting scope is difficult and expensive. ■ The software is developed during the implementation phase, so no early prototypes of the software are produced. ■ The model doesn't provide a clear path for problems found during testing phases. ■ Costly and required more time, in addition to detailed plan

Prototyping Model

■ Activity of creating prototypes
  ▷ (Throwaway prototyping)

Advantages	Disadvantages
■ Reduced time and costs, but this can be a disadvantage if the developer loses time in developing the prototypes. ■ Improved and increased user involvement.	■ Insufficient analysis· User confusion of prototype and finished system. ■ Developer misunderstanding of user objectives. ■ Excessive development time of the prototype. ■ Expense of implementing prototyping

Spiral Model (SDM)

- Combines elements of both design and prototyping-in-stages
- Features from prototyping model and the waterfall model
- Favored for large, expensive, and complicated projects

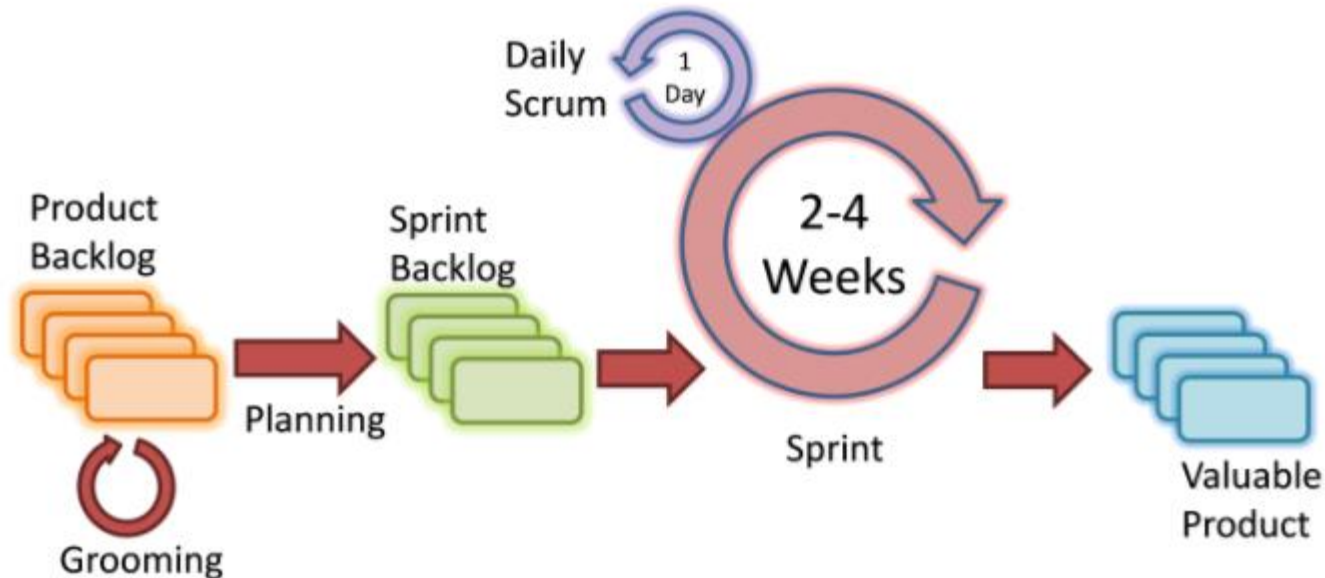| Advantages	Disadvantages
■ Estimates (i.e. budget, schedule, etc.) become more realistic as work progressed because important issues are discovered earlier. ■ Early involvement of developers. ■ Manages risks and develops the system into phases.	■ High cost and time to reach the final product. ■ Needs special skills to evaluate the risks and assumptions. ■ Highly customized limiting re-usability

Iterative and Incremental Model

■ Starts with an initial planning and ends with deployment with the cyclic interactions in between

Advantages	Disadvantages
■ Produces business value early in the development lifecycle.	■ Requires heavy documentation.
■ Better use of scarce resources through proper increment definition.	■ Follows a defined set of processes.
■ Can accommodate some change requests between increments.	■ Defines increments based on function and feature dependencies.
■ More focused on customer value than the linear approaches.	■ Requires more customer involvement than the linear approaches.
■ Problems can be detected earlier.	■ Partitioning the functions and features might be problematic.
	■ Integration between iteration can be an issue if this is not considered during the development.

Agile Model

- Based on iterative and incremental development, where requirements and solutions evolve through collaboration between cross-functional teams

| Advantages	Disadvantages
■ Decrease the time required to avail some system features. ■ Face to face communication and continuous inputs from customer representative leaves no space for guesswork. ■ The end result is the high-quality software in the least possible time duration and satisfied customer.	■ Scalability. ■ The ability and collaboration of the customer to express user needs. ■ Documentation is done at later stages. ■ Reduce the usability of components. ■ Needs special skills for the team.

Summary

- Clear idea of Processes in Object Orientated Analysis and Design
- Iterative and incremental development lifecycle
- Software development process framework (Software Development Lifecycle)
  - ▷ waterfall, iterative, agile, plan-driven, and so on

This Week

- Review Slides

- Read Chapter 7

- Online Quizzes

- Crossword Challenge

Crossword Challenge
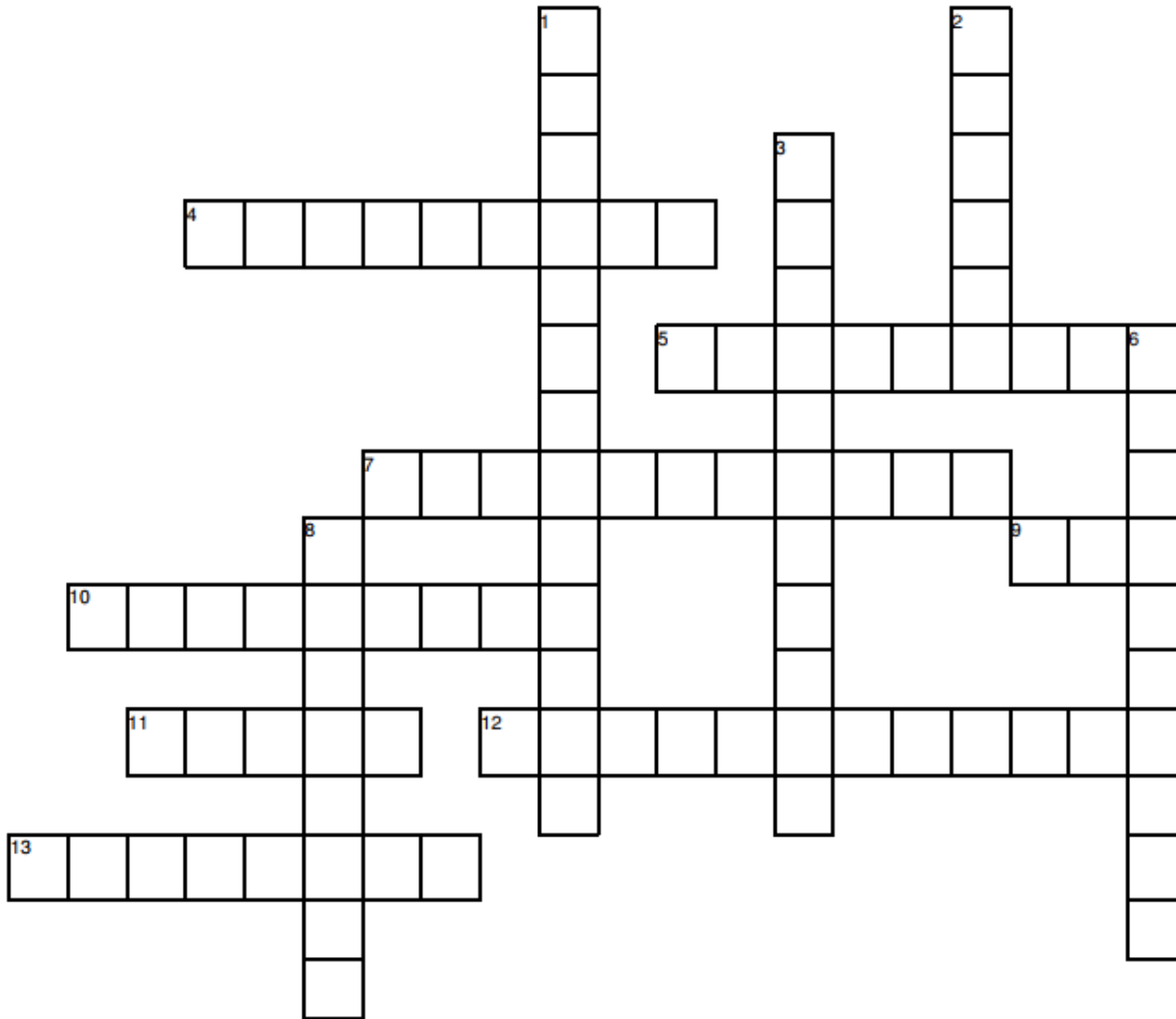
Across

4. A collection of operations that have no implementation but specify a particular service of a class or a component

5. Visibility access only by operations within the class or within children of the class

7. Classes specialized so their instances can manage a collection of other objects

9. An open extensible industry standard visual modelling language

10. A modular and replaceable part of a system that encapsulates its contents

11. A complete description of a system from a particular perspective

12. Different classes having methods of the same name that perform the same conceptual task

13. The degree to which classes within our system are dependent on each other

Down

1. The data inside an object is hidden and can only be manipulated by invoking one of the object's function

2. Visibility access by any element that can access the class

3. When classes are connected together conceptually

6. Diagram that maps system software to artifacts to the physical hardware that executes them

8. The measure of how much an entity supports a singular purpose within a system

Questions/Discussion

Question

- Having the most suitable process model will justify success of the project

  (Explain your answer)

a) True
b) False

Answer

- b) False

- There are a lot of factors that need to be considered for a successful software project, for example, requirement analysis is the most critical phase of software life-cycle, the skills of the project team and project manager, the quality of the deliverables, the used technologies, and so forth