Unit-1

Introduction to Software Design and Architecture

Overview

1.1 Overview of software design and architecture

1.2 Design levels: Architectural vs. detailed design

1.3 Design principles (modularity, abstraction, separation of concerns)

1.4 The role of the software architect

1.5 Key design goals: Performance, scalability, maintainability

1.6 Relationship between software design and software architecture

1.7 Object Oriented Software Development: Unified Software Development Process.

What is Architecture?

1: the art or science of building *specifically* **:** the art or practice of designing and building structures and especially habitable ones

2: **a:** formation or construction resulting from or as if from a conscious act
b: a unifying or coherent form or structure

3 : architectural product or work

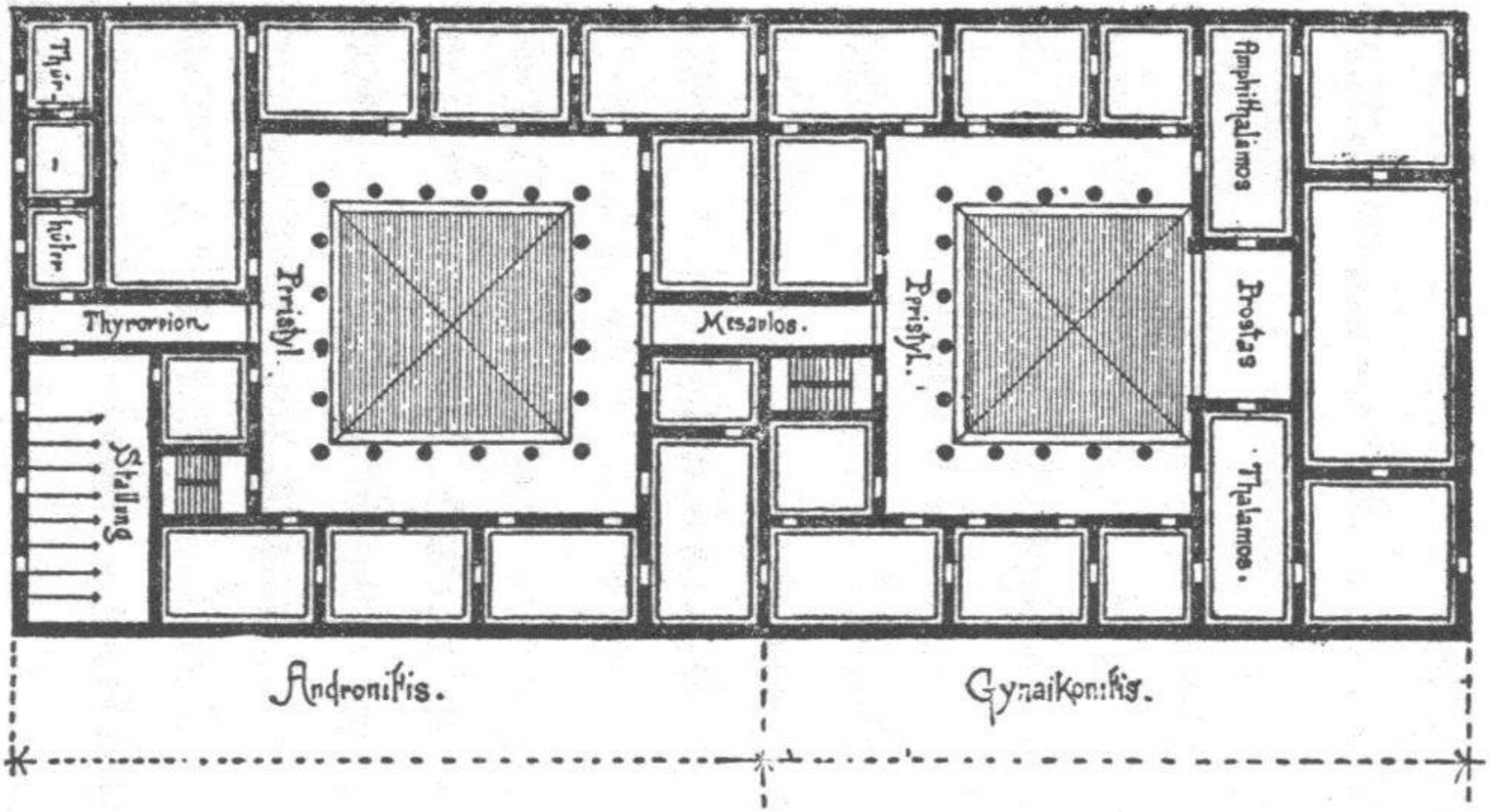
4 : a method or style of building

5 : the manner in which the components of a computer or computer system are organized and integrated

Vitruvius - *De architectura*

- **Marcus Vitruvius Pollio** (born c. 80-70 BC, died after c. 15 BC)
- author of **De Architectura**, known today as The Ten Books on Architecture
- carefully described existing practices, not only in the design and construction of buildings, but also in what are today thought of as engineering disciplines
- Vitruvius is famous for asserting that a structure must exhibit the three qualities of firmitas, utilitas, venustas - that is, it must be strong or durable, useful, and beautiful.

Roman house plan after Vitruvius



Mimar Sinan

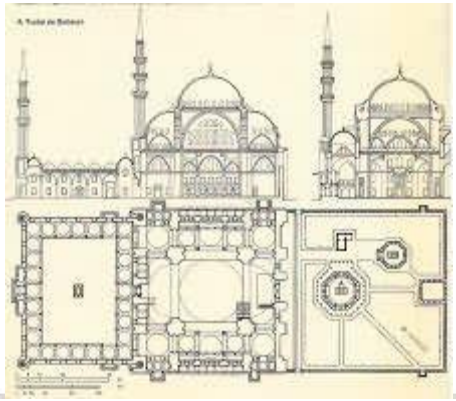
Sinan's major works

Şehzade Mosque in Istanbul

Şehzade Mosque (interior)

Süleymaniye Mosque in Istanbul

Süleymaniye Mosque (interior)



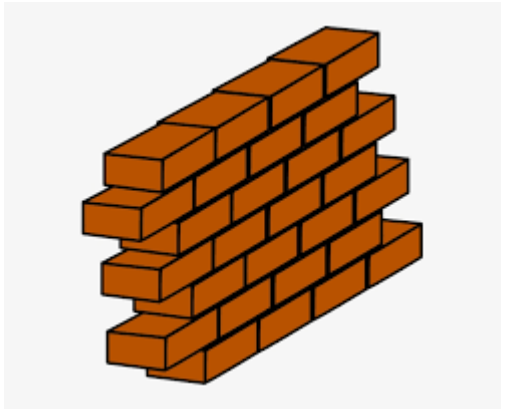
Selimiye Mosque

More Architectures

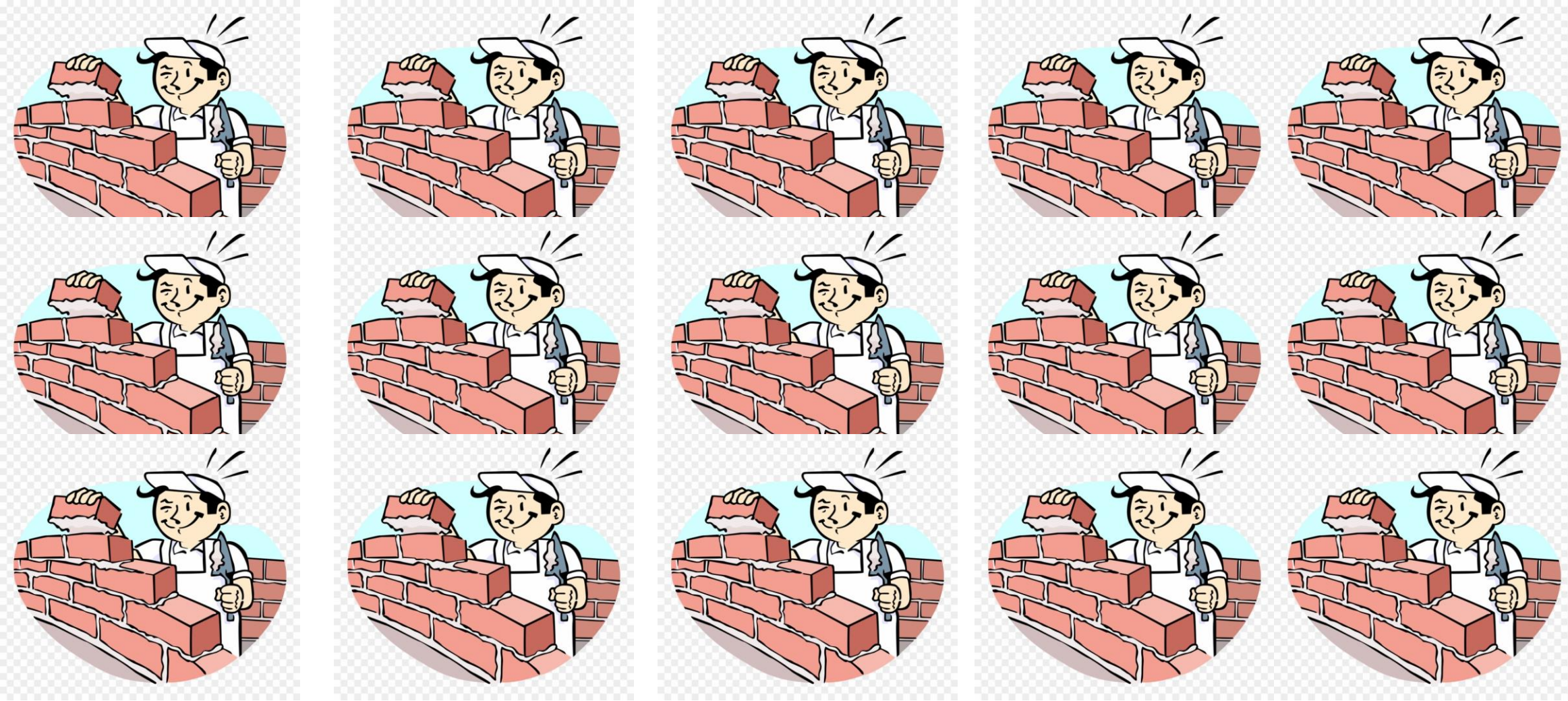
Taj Mahal

Build a Palace

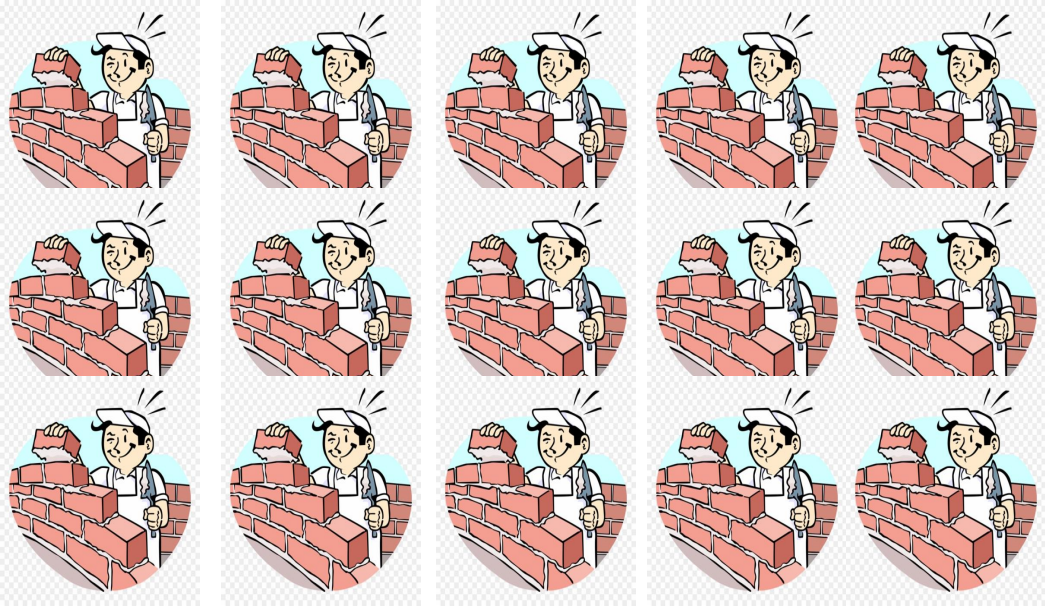
- Need a Palace…
- Get some bricks, wood…
- Go find many masons, carpenters as you can.
- Build the palace…?



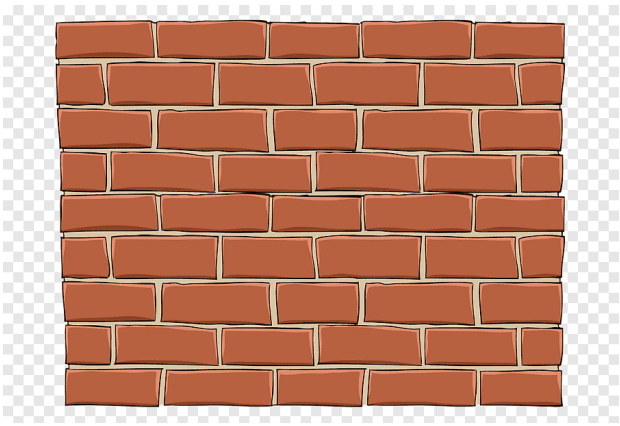
Build a Palace?



Build a Palace?



?



You Need a Architect First…

Architecture

Architecture Realization

Real Building

Software Development

- Lists
- Arrays
- Class
- Object
- Procedures
- Functions
- Algorithms
- Etc.

WebSpider.py - C:\Python32\WebSpider.py

File Edit Format Run Options Windows Help

```
from html.parser import HTMLParser
from urllib.request import urlopen
from urllib import parse

class LinkParser(HTMLParser):

    def handle_starttag(self, tag, attrs):
        if tag == 'a':
            for (key, value) in attrs:
                if key == 'href':
                    newUrl = parse.urljoin(self.baseUrl, value)
                    self.links = self.links + [newUrl]

    def getLinks(self, url):
        self.links = []
        self.baseUrl = url
        response = urlopen(url)
        if response.getheader('Content-Type')=='text/html':
            htmlBytes = response.read()
            htmlString = htmlBytes.decode("utf-8")
            self.feed(htmlString)
            return htmlString, self.links
        else:
            return "",[]

def spider(url, word, maxPages):
    pagesToVisit = [url]
    numberVisited = 0
    foundWord = False
    while numberVisited < maxPages and pagesToVisit != [] and not foundWord:
        numberVisited = numberVisited +1
        url = pagesToVisit[0]
        pagesToVisit = pagesToVisit[1:]
        try:
            print(numberVisited, "Visiting:", url)
            parser = LinkParser()
            data, links = parser.getLinks(url)
            if data.find(word)>-1:
                foundWord = True
            pagesToVisit = pagesToVisit + links
            print(" **Success!**")
        except:
            print(" **Failed!**")
    if foundWord:
        print("The word", word, "was found at", url)
    else:
        print("Word never found")
```

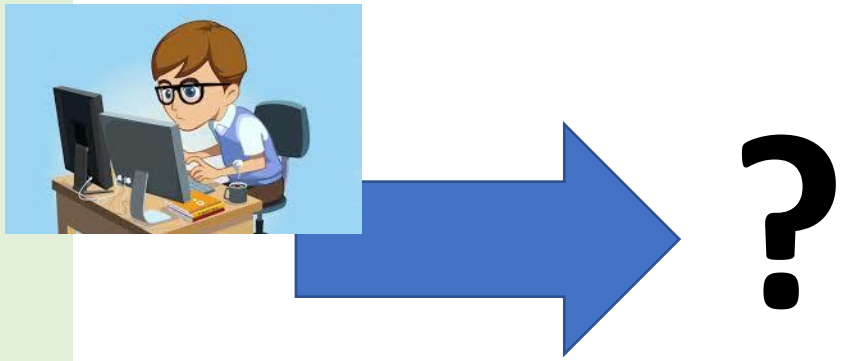
Ln: 47 Col: 33

Large-Scale, Complex software Systems…

- Large(Distributed) system
- Many people working on the same problem
- Overly Complex
- Millions of Code…
- Should be delivered on time and within budget!

Coding only will not do…

- Lists
- Arrays
- Class
- Object
- Procedures
- Functions
- Algorithms
- Etc.



Adding More Programmers… ??





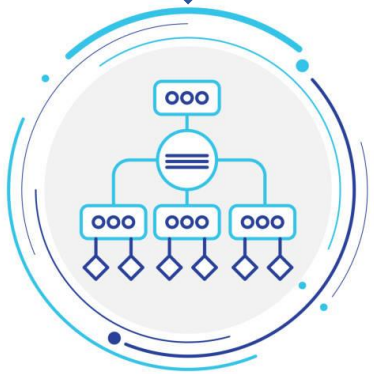

Mythical Man-Month

- Fred Brooks, The Mythical Man-Month(1975)
- “Adding manpower to a late software project makes it later.”

- **Conceptual Integrity**
  - To make a user-friendly system, the system must have conceptual integrity, which can only be achieved by separating architecture from implementation.
  - A single chief architect... decides what goes in the system and what stays out.
  - "Having a system architect is the most important single step toward conceptual integrity...after teaching a software engineering laboratory more than 20 times, I came to insist that student teams as small as four people choose a manager, and a separate architect."

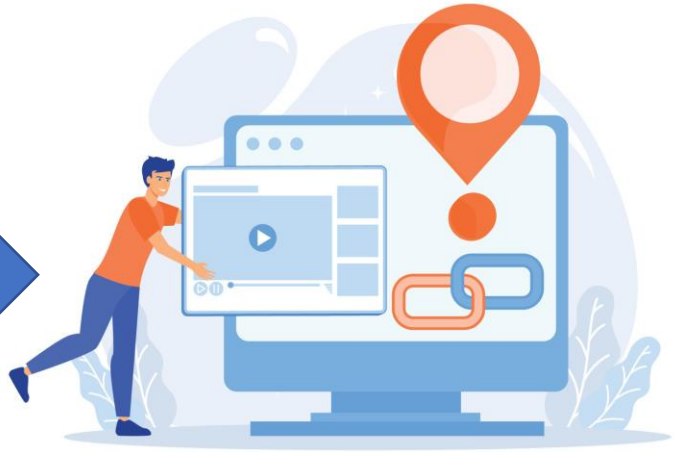
A Software Architect?

Software Architecture Design



Software Architecture

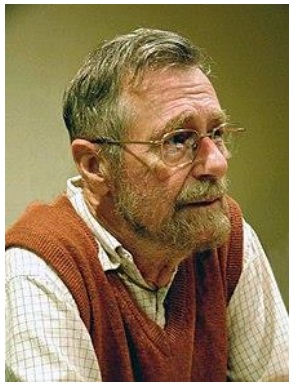
Software Architecture Realization



Application

The Architect…

Structure Matters - Dijkstra 1968



Edsger W. Djikstra
1930-2002 A.D.

- **Dijkstra, 1968**
  - "...Correct arrangement of the structure of software systems before simple programming..."
- **Layered Structure**
  - Programs are grouped into layers
  - Programs in one layer can only communicate with programs in adjoining layers
- With easier development and maintenance

Focused on Structure of Programming

Structure Matters - Parnas 1972

- "...selected criteria for the decomposition of the system impact the structure of the programs and several **design principles** must be followed to provide a **good structure**..."
- **Information-hiding modules (1972)**
  - ❑ Identify design decisions that are likely to change
  - ❑ Isolate these in separate modules (separation of concerns)
- **Software Structures(1974)**
  - ❑ Hierarchical structures (stepwise refinement) in programs
- **Program Families (1975)**
  - ❑"A program family is a set of programs for which it is profitable or useful to consider as a group."

Structure in Software

	Structure in Software	PROBLEMS SOLVED
2010	APPROACH Programming-the-world-software architecture	Mega programs
2000 1990	Programming-the-large -object-oriented design - CASE tools -libraries	Large, complex, distributed
1970	Programming-the-small -information hiding, modularization	Data intensive, business applications
1960 1950	Programming any-which-way	Simple, algorithmic

'Structure' Concept in Software

1960s - Structured Programming

- Adopted into programming languages because its a better way to think about programming

1970s- Structured Design

- Methodology/guidelines for dividing programs into subroutines

1980s- Modular programming languages

- Modular (object-based) programming

Grouping of sub-routines into modules with data

1990s- Towards Software Architectures

- Object-Oriented Analysis/Design/Programming started being commonly used
- Software Architecture Design

2000s - Software Architectures

- Starting to be Common Practice
- Specialization of concepts

Architecture…

- Building Architecture
- Car Architecture
- Computer Architecture
- Plane Architecture
- Bridge Architecture
- …

Architecture definition from a class…

- Your own definition for architecture…

Architectural Structures and Views

- Architectural structures have counterparts in nature. For example, the neurologist, the orthopedist, the hematologist, and the dermatologist all have different views of the various structures of a human body, as illustrated in Figure 1.1. Ophthalmologists, cardiologists, and podiatrists concentrate on specific subsystems. Kinesiologists and psychiatrists are concerned with different aspects of the entire arrangement's behavior. Although these views are pictured differently and have very different properties, all are inherently related and interconnected: Together they describe the architecture of the human body.

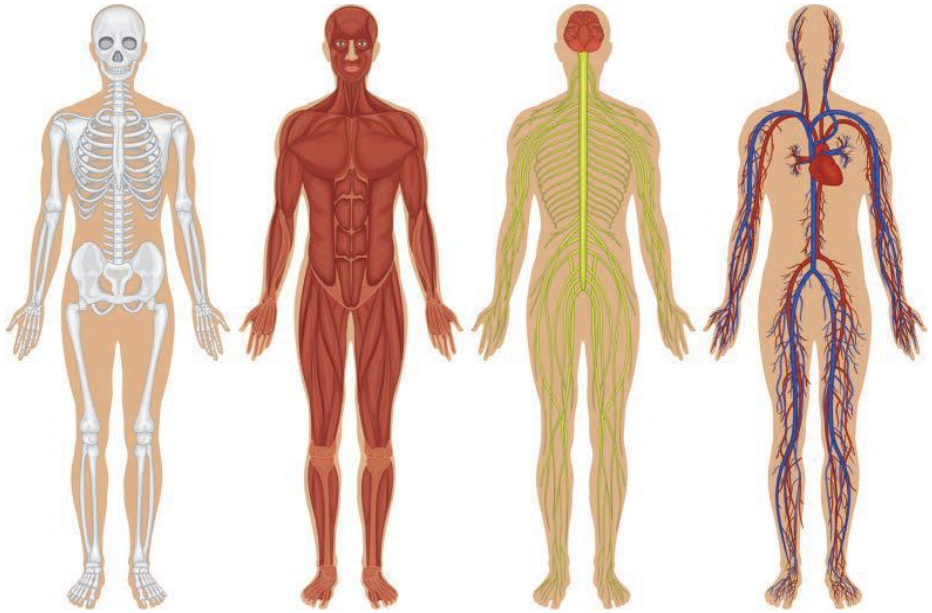


FIGURE 1.1 Physiological structures

Software Architecture – Booch 1991

- *"The logical and physical structure of a system, forged by all the strategic and tactical design applied during development."*

Architecture is a high level structure of the software system

Software Architecture- Garlan and Perry 1995

- *“The structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time.”*

Discussion…

- What is the rationale for software architecture according to you?

Software Architect…

- Who is a software Architect?

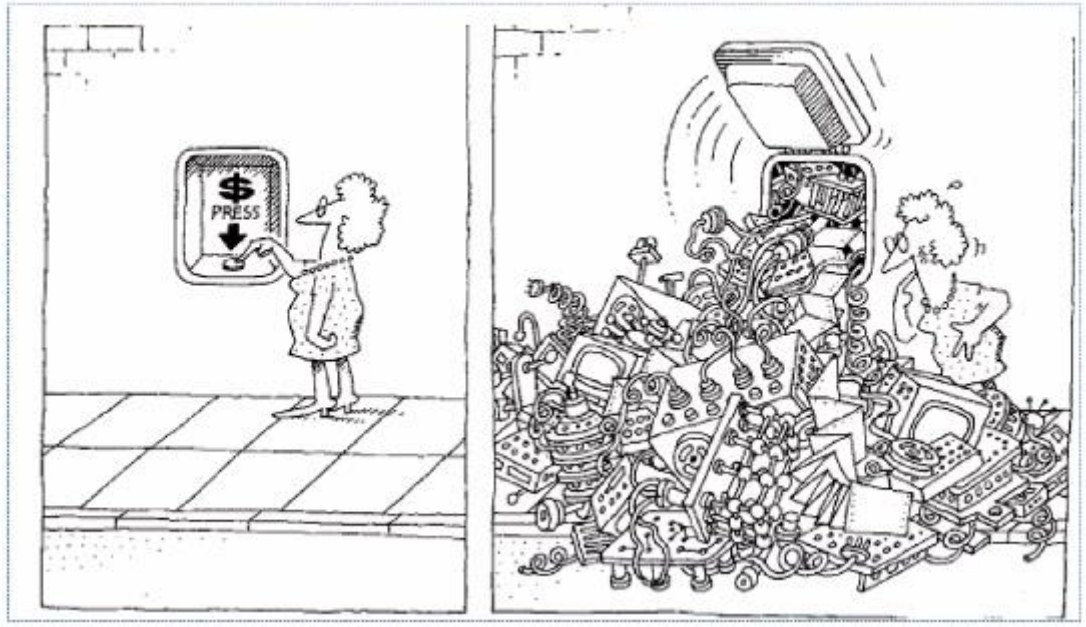
You know him?

William(Bill) H. Gates,

Chief Software Architect, Microsoft

1 – Abstract Specification

- Abstraction
  - Focus only on relevant properties of the problem
  - ‘Ignore’ details



Abstract Specification

- Engine combine to chassis
- Engine rotates wheels
- Chassis carries wheels
- Chassis carries Brakes
- Brakes stops the wheels
- …

Abstract Specification

- Architecture represents a **high level abstract specification**
- Abstraction helps to cope with **complexity**
- Abstraction **improves understanding** of the software system

2 – Stakeholder Communication

- Stakeholder is any person who has interest in the architecture
- Stakeholders:
  - End users
  - Managers
  - Developers
  - Maintainers
  - Analysts
  - Designers
  - …

2 – Stakeholder Communication

- Software architecture provides a common medium for communication among stakeholders
- This will improve understanding of the system among stakeholders

3 – Coping with Evolution

- About 80% cost of the software system occurs after initial deployment
- Software systems change over their lifetimes. Often!
- Changes can be categorized into:
  - Local: change to a single element
  - Non-Local: change to multiple elements but leaves architecture intact
  - Architectural: change is systemic, and affects the overall structure

3 – Coping with Evolution

- Architecture can help dealing with changes and evolution.
- In case of proper architecture definition, the changes will be limited to the abstraction boundaries
- Architecture provides the balance between fixed and adaptable parts of the system.

4 – Guides Software Development Process

- Architecture is explicit
- Focus on Architectural Components
- Analyze and design based on the architectural Components.

5 – Organization of the Development project

- The architecture influences the organizational structure for development/maintenance efforts.

- **Examples include:**
  - division into teams
  - units for budgeting, planning
  - basis of work breakdown structure
  - organization of documentation
  - basis of integration
  - basis of test plans, testing
  - basis of maintenance
  - Incremental deployment

6 - Large Scale Reuse

- Software Architecture is an abstract specification
- Representing set of systems
- And as such can be reused for systems exhibiting similar structure and requirements
- Can promote software product lines

Rationale for Software Architecture

1. Improved understanding because of a higher level abstract specification.
2. Vehicle for communication among different stakeholders because of common abstract specification
3. Manifestation of the earliest design decisions
4. Guides software development process
5. Supports organization of development project
6. Provides gross level reuse.

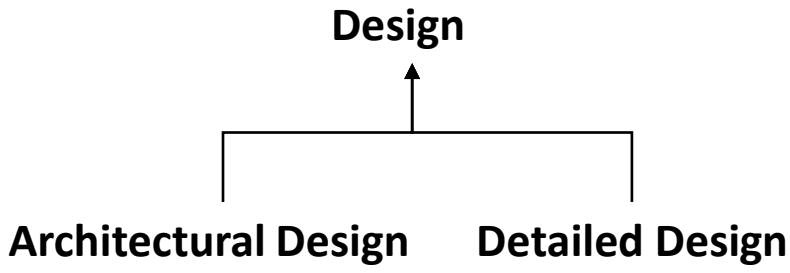
Common misconception…

Architecture is just paper…

- Every system has an architecture: either visible or not
- The architecture eventually resides in executable code
- A system’s Architecture may be visualized in models; which can be executable

Architecture vs Design

- All architecture is design, but not all design is architecture (e.g. detailed design)
- Architecture focuses on significant design decisions, decisions that are both structurally and behaviorally imported as well as those that have lasting impact on the performance, reliability, cost and resilience of the system.
- Architecture is at a higher abstraction level



Software Design

- Software design refers to the process of defining the architecture, components, interfaces, and other characteristics of a system.
- It involves the application of principles and methods to develop high-quality software solutions.
- Design is crucial for addressing complexity, ensuring scalability, maintainability, and enabling clear communication among team members.

Software Architecture

- Software architecture refers to the high-level structure of a system, focusing on how different components or modules interact.
- It establishes the foundation for both functional and non-functional requirements, such as performance, security, and maintainability.
- It is a blueprint that guides the detailed design and development process.

software design and architecture

- When we talk about **"software architecture for developers"**, we're referring to the high-level structuring of software. Think of it as the **blueprint for system design, much like an architect's plan for a building**. It lays out the big picture, guiding how different parts of a software system will come together.

- On the other hand, software design dives into the nitty-gritty. It's all about defining the methods, functions, objects, and how these objects interact with each other. While software architecture management ensures that the blueprint is followed and evolves as needed, **software design ensures that every brick (or line of code) fits perfectly in its place.**

1. Architectural Design

- **Purpose**: To define the overall system structure and guide the development process.
- **Key Elements**:
  - Identifies the main modules or components of the system.
  - Specifies how components interact with each other (e.g., through APIs, messaging systems).
  - Considers high-level concerns like scalability, performance, security, and technology stack.
- **Typical Outputs**:
  - Architecture diagrams (e.g., layered diagram, component diagram).
  - System blueprints that define the responsibilities of each module and how they integrate.

2. Detailed Design

- **Purpose**: To describe the internal details of each system component or module.
- **Key Elements**:
  - Defines the classes, methods, and data structures within each module.
  - Specifies the algorithms, workflows, and object interactions.
  - Focuses on implementation details and code structure, ensuring that each part functions as intended.
- **Typical Outputs**:
  - Class diagrams, sequence diagrams, data flow diagrams.
  - Pseudocode or detailed specifications for developers.

Software Development Activities

- Requirement Elicitation
- Requirement Analysis
  - Analyzing requirements and working towards a conceptual model without taking the target implementation technology into account
  - Useful if the conceptual gap between requirements and implementation is large
- Design
  - Coming up with solution models taking the target implementation technology into account
- Implementation
- Test

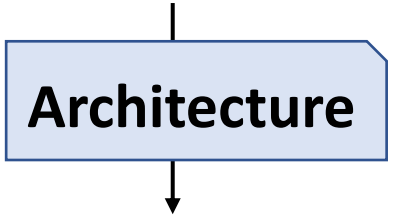
Waterfall Model

1. Requirement Elicitation	**What? (Client)**
2. Analysis	**What? (Domain)**
3. Design	**How? (Detailed)**
4. Implementation	**Do**
5. Testing	**Test**
6. maintenance	**maintain**

Where is Architecture Design?

Waterfall Model

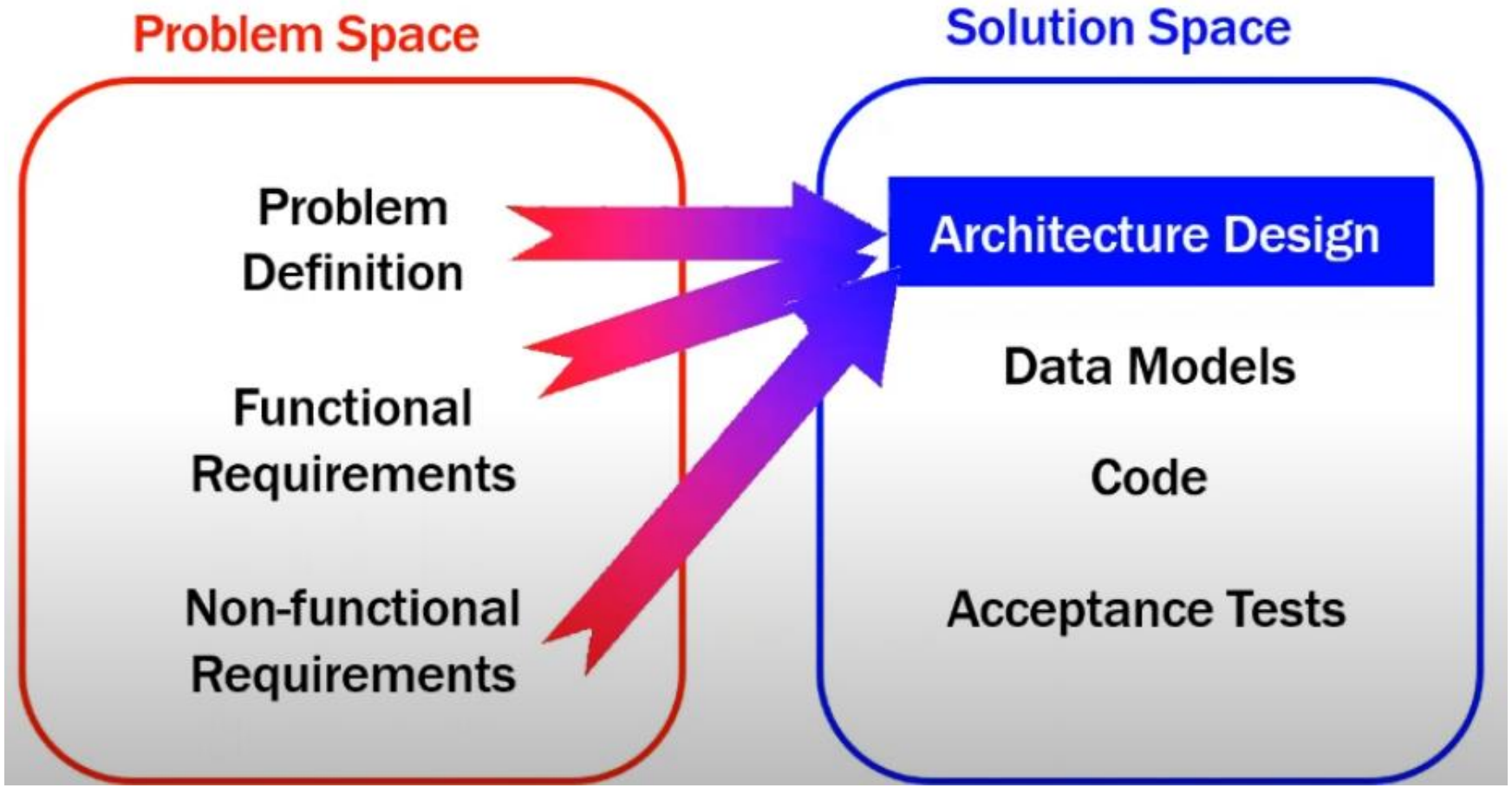
1. Requirement Elicitation — **What? (Client)**



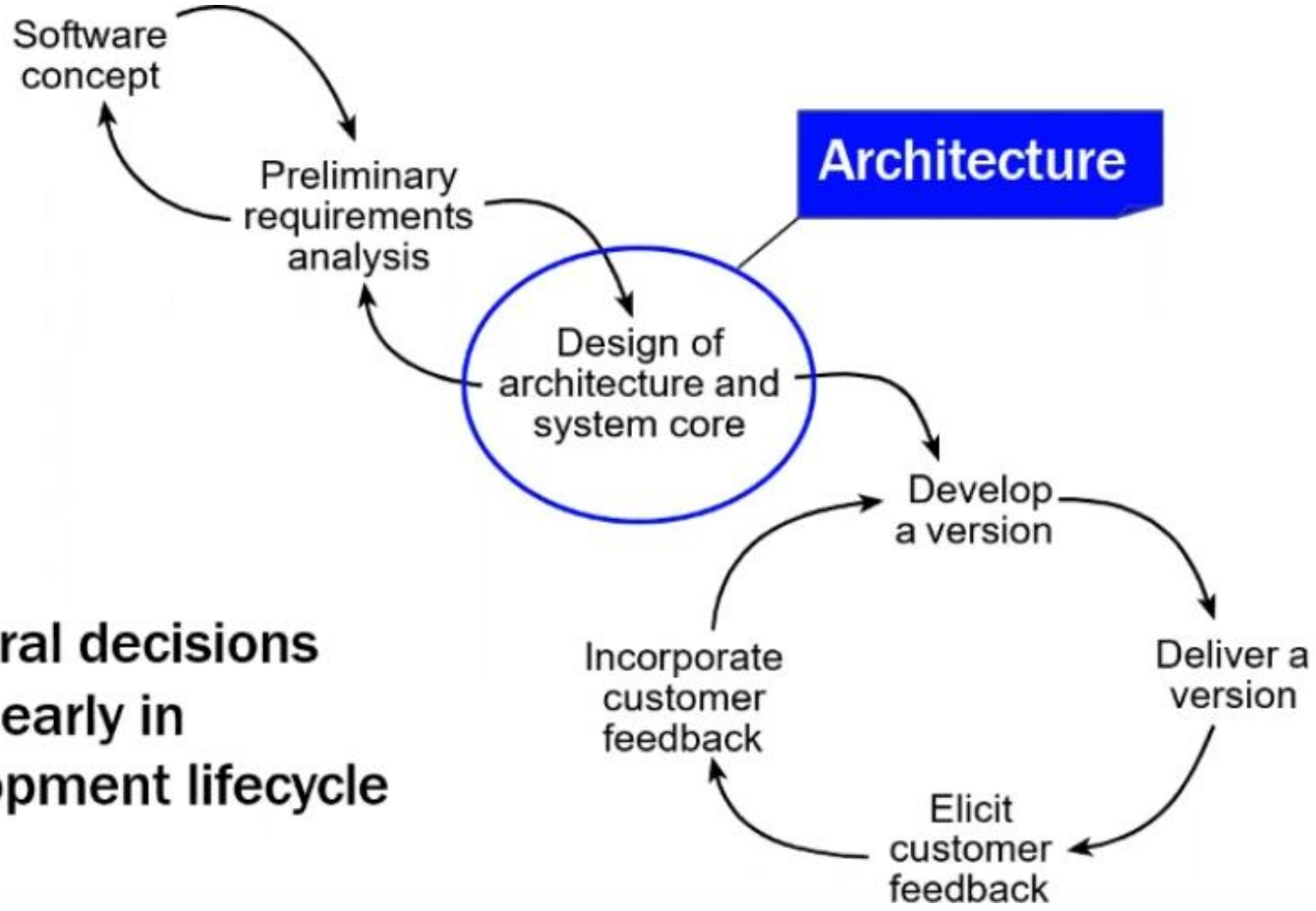
What? (Domain, gross-level)

2. Analysis — **What? (Domain)**
3. Design — **How? (Detailed)**
4. Implementation — **Do**
5. Testing — **Test**
6. maintenance — **maintain**

Bridge the Gap



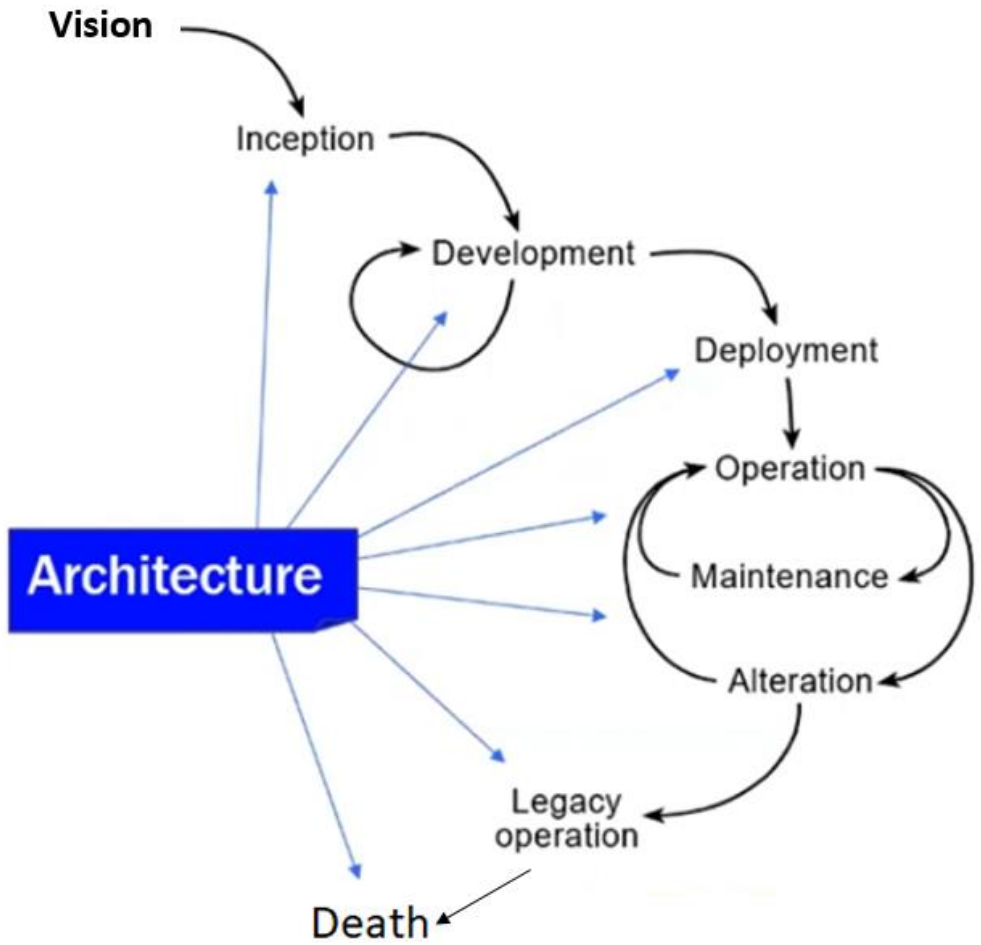
Evolutionary Model



Architectural decisions are made early in the development lifecycle

System Lifecycle

Architectural decisions affect the whole lifetime of a system



Design Goals

Good Design Leads to software that is:

1. **Correct** – does what it should
2. **Robust** – tolerant of misuse, e.g. faulty data
3. **Flexible** – adaptable to shifting requirements
4. **Reusable** – cut production costs for code
5. **Efficient** – good use of processor and memory

Also should be **Reliable** and **Usable**

Key Design goals

- Performance
- Scalability
- Maintainability

Key Design goals…

- **Performance**:
  - This focuses on ensuring the system runs efficiently and responds quickly to user interactions. It involves optimizing resource usage, such as memory, CPU, and network bandwidth, and may involve selecting efficient algorithms, minimizing data transfers, and reducing processing overhead.
  - Performance is crucial in applications where response time and speed directly impact user satisfaction and overall system effectiveness.

Key Design goals…

- **Scalability**:
  - Scalability is about designing systems that can handle an increase in workload (more users, larger data, etc.) without sacrificing performance. This often means the system can be expanded or upgraded to accommodate growth.
  - Techniques include load balancing, horizontal or vertical scaling, and partitioning data. Scalability ensures that as the application grows, it continues to function efficiently.

Key Design goals…

- **Maintainability**:
  - Maintainability refers to how easy it is to update, debug, or extend the system. This is key for long-term success, as systems often need new features, bug fixes, and optimizations over time.
  - Practices for maintainable design include clear code structure, modularity, proper documentation, and adherence to coding standards. Maintainable systems are more resilient to changes and easier to adapt.

Non Functional Requirements

Organizational

- Delivery (e.g. quality of documentation, manuals, training, support)
- Implementation (e.g. commenting, flexibility, openness, maintainability)
- Standards (e.g. ISO 9000)

External

- Legislative
  - Privacy
  - Safety
  - Commercial(e.g. License)
- Interoperability
- Ethical

Architecture and Design

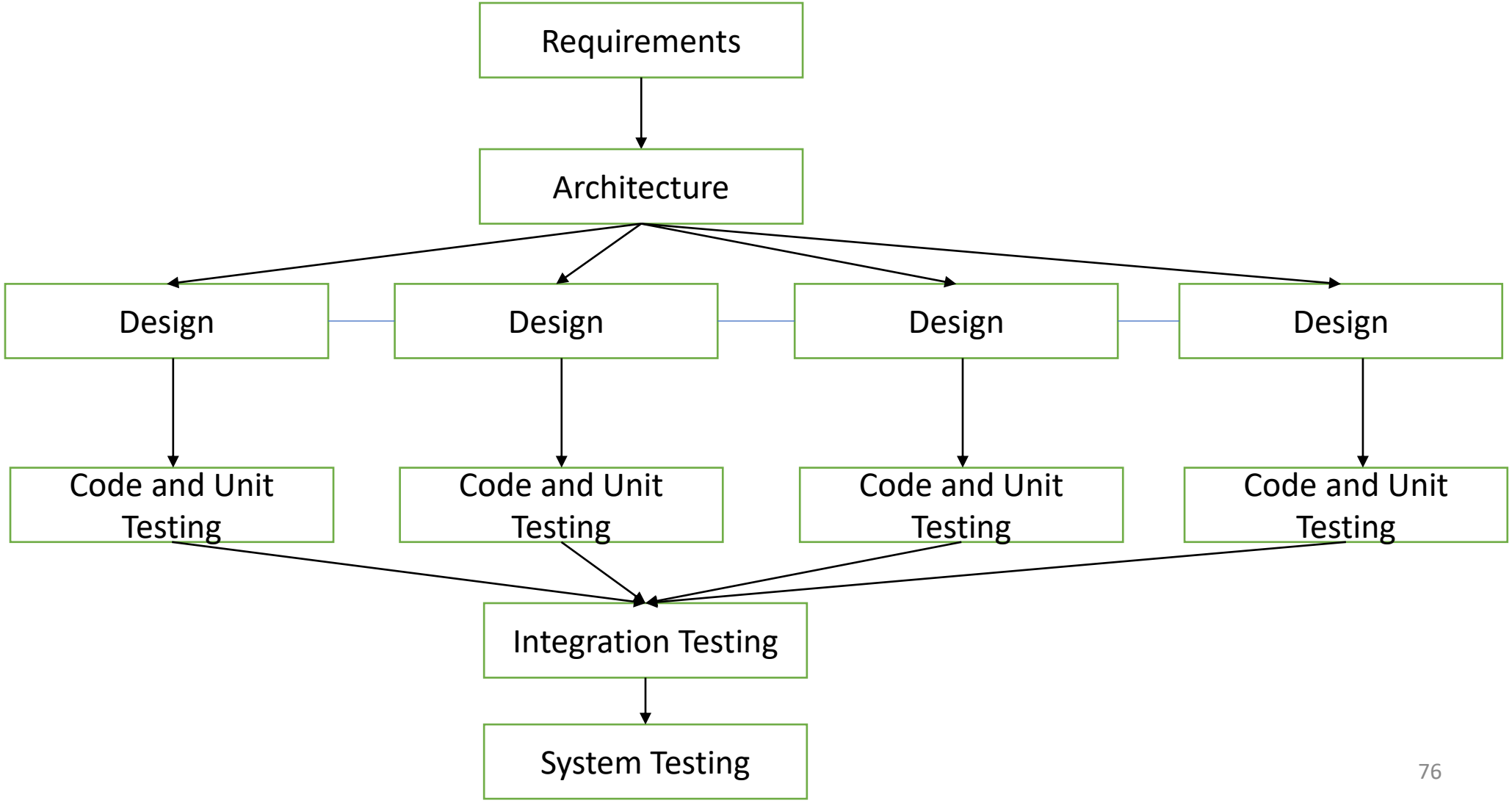
Architecture

- High-Level
- Major Decisions
- Not even thinking about programming

Design

- ***"laying out"*** the programming language code used to implement the architecture
- Organizing programming language concepts

Design and Architecture in the Development Process



Object Oriented Software Development

- The Object-Oriented Software Development (OOSD) process is an approach to designing and developing software by organizing it into objects that represent real-world entities or concepts.
- This process leverages the principles of Object-Oriented Programming (OOP), which include encapsulation, inheritance, polymorphism, and abstraction.
- The goal is to create software that is modular, reusable, and easier to maintain.
- The OOSD process is iterative, meaning it typically involves multiple cycles of refining and improving the software.
- It’s often combined with methodologies like the Unified Software Development Process, which includes incremental and iterative development

Phases of OOAD

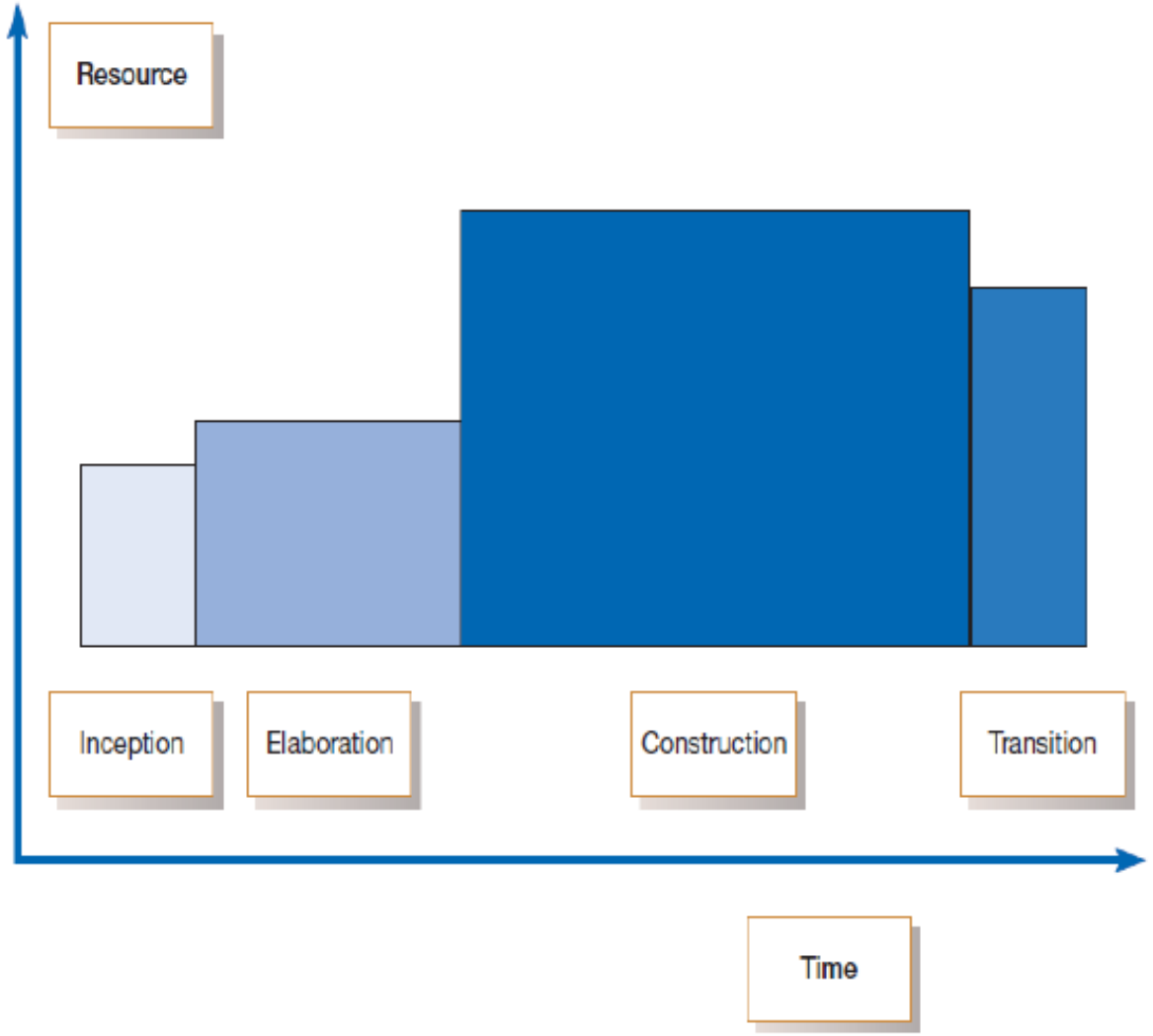
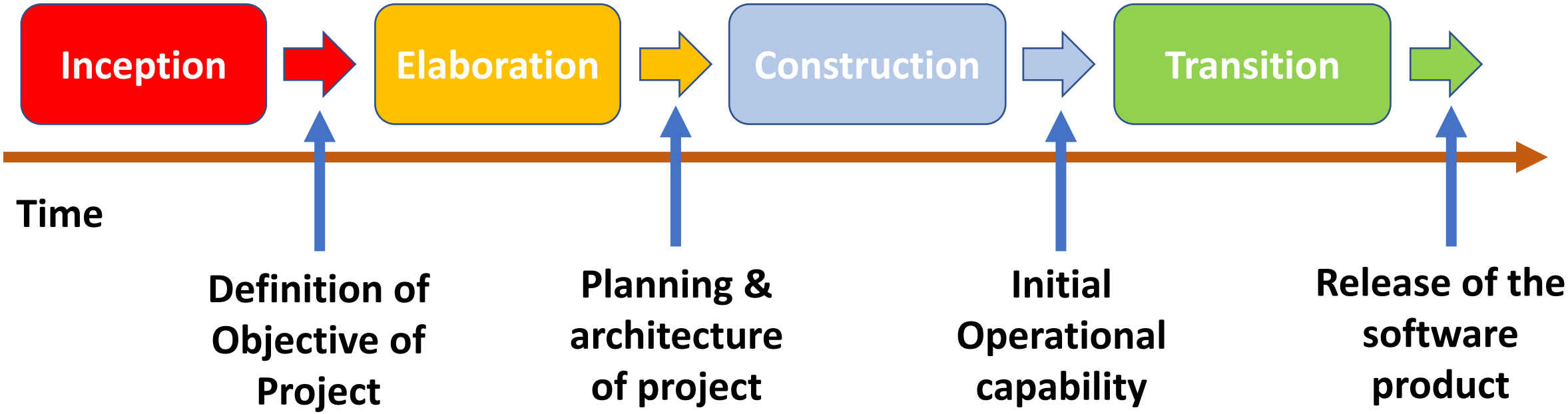


FIGURE 1-11
Phases of OOAD-based development

Unified Process Model



Phases of Unified Process

- The software life cycle is divided into cycle, each cycle working on a new generation of the product. This process divide one cycle into four phases and each phases include well-define milestone.

1. Inception phase:

- In this phase business cases are identified and established for the system which includes success criteria, risk assessment, estimates of the resource needed, preparation of development environment, analysis of critical non-functional requirements, what function to include and what to exclude and phase plan showing dates of milestone.
- Inception is not a requirement phase rather it is a feasibility phase where enough investigation is done to support a decision to continue or stop. The milestone in this phase is a **"lifecycle objective".**

Phases of Unified Process

- The indication that project has reached this milestone includes:
  - The major stake holder agrees on the scope of the proposed system.
  - If a set of critical high level requirements are addressed by business scope.
  - The business case of the project is strong enough to continue the development.

- Most requirements analysis are done in elaboration phase so the inception phase should be relatively short for most project such as one or few week long. This process also define what process to use and what tools to use.

How much UML during Inception:

- The purpose of inception phase is to collect just enough information to establish a common vision, decide if moving forward is feasible or not. So beyond simple use case diagram, not much diagram is warranted. But only 10% of uses case are only analyzed deeply.

Phases of Unified Process

2. Elaboration phase:

- In this process the problem domain is analyzed, sound architectural structure are established, project plan are developed, high risk elements of the project are eliminated. Architectural design have to be made with the understanding of system's scope, major functionality, non-functional requirement like performance requirement, accuracy, speed of operation etc.
- This phase involves refinement of vision, iterative implementation of the core architecture, resolution of high risk, identification of more requirement and scope.
- It is the initial series of iteration during which the team does the serious investigation, implements (program and test) the core architecture and tackle the high risk issues. Early work may include implementing scenarios that are deemed important but are not technically risky. Here prototype is not created rather code and design are consider as portion of the final system. Executable architectural prototype is built in one or more iteration which means a production subset of the final system.

Phases of Unified Process

- The major milestone associated with the elaboration phase is "**Lifecycle Architecture".** The indication that the project have reached this milestone include:
  - Most of the functional requirement has been capture in use case model.
  - The architecture baseline is small.
  - Project team has initial project plan that describe how the construction phase will proceed and if business case has received green signal.

Phases of Unified Process

3. Construction phase:

- This phase is consider as the largest phase in the project which is concerned with the system design, programming and testing. The main goal is to build the system capable of operating successfully in beta customer environment. The project team perform the task that involves building the system iteratively and incrementally making sure that the system is always evident in executable form. All the component and application feature are developed and integrated into the product and are thoroughly tested. On completion of this phase we have a working software system.
- The major milestone associated with this phase is "**Initial Operational Capability**" which indicates whether the project is ready to be deployed in beta test environment.

Phases of Unified Process

- The indication that the project have reached this milestone include:
  - If a set of beta customers has a more or less fully operational system in their hands.
  - If the product release is stable and mature enough to be deployed.
  - If actual resource expenditure vs standard resource expenditure is acceptable

Phases of Unified Process

4. Transition Phase:

- It is the final phase of the process and its purpose is to transit the final software product to customer environment. The project team focuses on correcting the defect and modifying the system to correct previously unidentified problem. It also includes system conversion and user training. Feedback received from the initial release includes further refinement to be incorporated.
- The major milestone associated with the transition phase is "**product release**" which indicates whether the objectives are met or not and start plan for new development. The indication that the project have reached this milestone include:
  - If the users are satisfied
  - If actual resource expenditure vs standard resource expenditure is still acceptable.

Why to use Unified Process (UP)?

1. Support iterative and incremental development
2. Manage Requirement
3. Use case driven
4. Architecture centric
5. Control change to software
6. Continuously verify software quality