SE 3112

SOFTWARE ARCHITECTURE

Prepared by: Dams Gabriel Lazarus
ICT University, Fall 2022

CHAPTER 1:

Introduction to Software Architecture

Topics Overview

- Understanding the Concept of Software Architecture
- What is Software Architecture
- Rationale for Software Architecture
- Common Misconceptions about Software Architecture
- Software Architecture in the Life Cycle
- Influences on Software Architecture

What is Architecture?

- Webster's definition of Architecture
  - The art or science of building:
  - The art or practice of **designing and building structures** and especially habitable ones
  - **A unifying or coherent form or structure**
  - Architectural **product** or work
  - A **method** or **style** of building
  - The **manner** in which computer components/systems are organized and integrated.

Vitruvius – De Architectura

- Marcus Vitruvius Pollio (born c. 80-70 BC, died after 15 BC).

- Author of De Architectura, known today as the Ten Books of Architecture
  - Carefully described existing practices, not only in the design 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.

Building Architectures

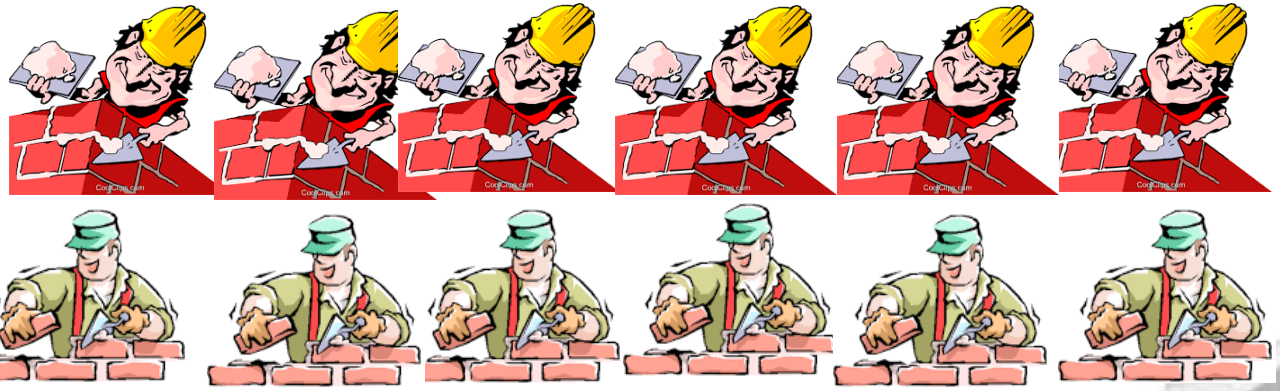
Petronas Twin Towers Malaysia

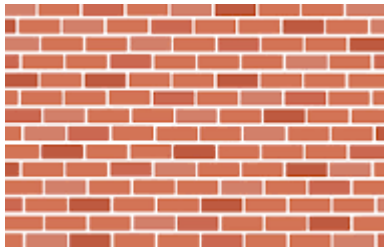
Petronas Twin Towers or KLCC Twin Towers, are 88-storey supertall skyscrapers in Kuala Lumpur, Malaysia

Build Twin Towers ....

- Imagine you need twin towers
- Get some bricks, wood…
- Go find many masons, carpenters as you can.
- Build the towers…?
- Is it OK?

Build Twin Towers ....





You Need an Architect first...

Architecture Design

Architecture

Architecture Realization

Building

Software Development

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

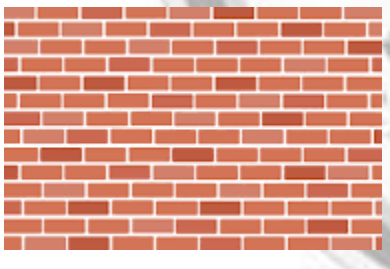
```
<!-- todo: put this in a different file!!! -->
<script>
function authenticateUser(username, password) {
  var accounts = apiService.sql(
    "SELECT * FROM users"
  );

  for (var i = 0; i < accounts.length; i++) {
    var account = accounts[i];
    if (account.username === username &&
        account.password === password)
    {
      return true;
    }
  }
  if ("true" === "true") {
    return false;
  }
}

$('#login').click(function() {
  var username = $("#username").val();
  var password = $("#password").val();

  var authenticated = authenticateUser(username, password);

  if (authenticated === true) {
    $.cookie('loggedin', 'yes', { expires: 1 });
  } else if (authenticated === false) {
    $("#error_message").show();
  }
});
</script>
```



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

Programming in the Large

For these kinds of systems, architecture is a must.

Coding only will not do…

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


```
<!— todo: put this in a different file!!! —>
<script>
function authenticateUser(username, password) {
  var accounts = apiService.sql(
    "SELECT * FROM users"
  );

  for (var i = 0; i < accounts.length; i++) {
    var account = accounts[i];
    if (account.username === username &&
        account.password === password)
    {
      return true;
    }
  }
  if ("true" === "true") {
    return false;
  }
}

$('#login').click(function() {
  var username = $("#username").val();
  var password = $("#password").val();

  var authenticated = authenticateUser(username, password);

  if (authenticated === true) {
    $.cookie('loggedin', 'yes', { expires: 1 });
  } else if (authenticated === false) {
    $("#error_message").show();
  }
});
</script>
```

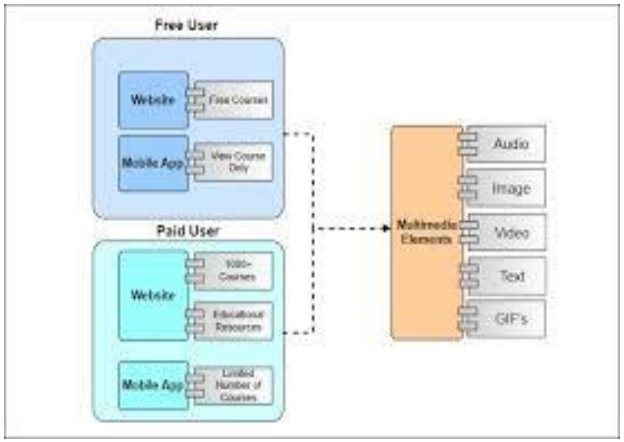
- You will get nice programs in coding.
- But for you to get the overall structure of the software, you need more coding.

Mythical Man-Month…

- Fred Brooks, The Mythical Man-Month (1975, 1st edition)
  - "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.
    - It is always advised to have a separate architect in a software development team.

A Software Architect

Architecture Design



Architecture

Architecture Realization

```
<!-- todo: put this in a different file!!! -->
<script>
function authenticateUser(username, password) {
  var accounts = apiService.sql(
    "SELECT * FROM users"
  );

  for (var i = 0; i < accounts.length; i++) {
    var account = accounts[i];
    if (account.username === username &&
        account.password === password)
    {
      return true;
    }
  }
  if ("true" === "true") {
    return false;
  }
}

$('#login').click(function() {
  var username = $("#username").val();
  var password = $("#password").val();

  var authenticated = authenticateUser(username, password);

  if (authenticated === true) {
    $.cookie('loggedin', 'yes', { expires: 1 });
  } else if (authenticated === false) {
    $("#error_message").show();
  }
});
</script>
```

Software Application

Structure Matters – Dijkstra 1968

- 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

- Why? To support easier development and maintenance

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**.."

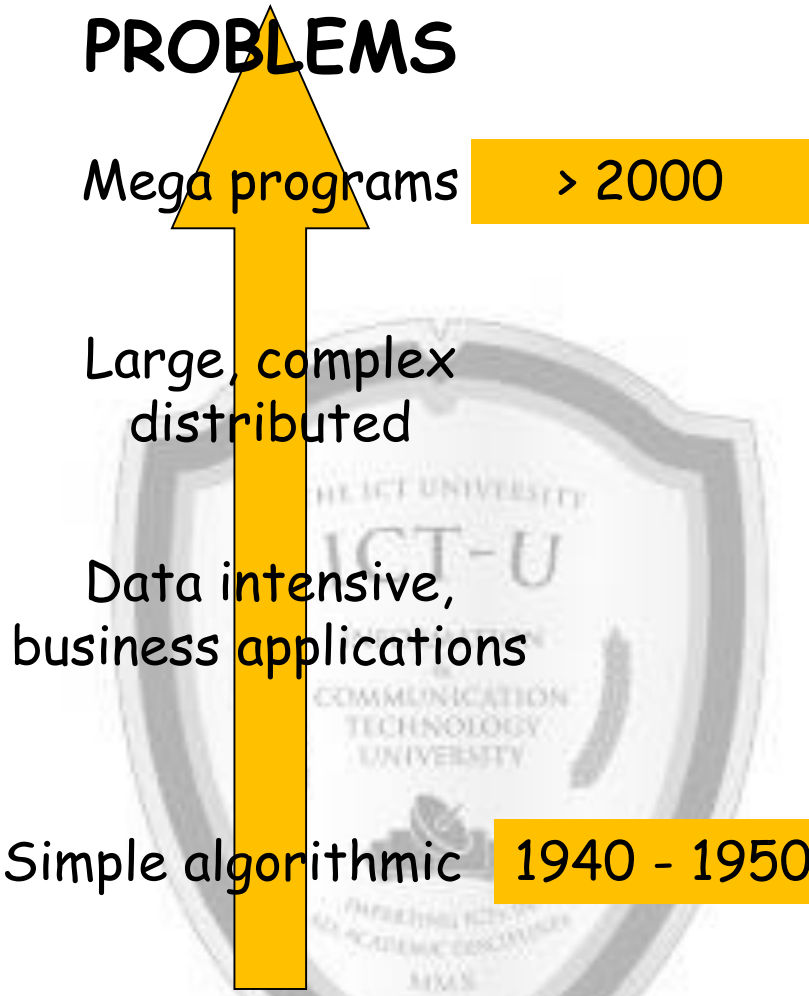
- Information-hiding modules (1972)
  - Identify design decisions that are likely to change
  - Isolate these in separate modules

- Software Structures (1974)
  - Hierarchical structures 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." - Reuse

Evolution of Software Problems

- Software problems have changed in nature (algorithmic to general-purpose applications)
- Increased in complexity (millions of lines of code)
- It has now become very hard/impossible to develop system without architecture.



Structure in Software

Years	APPROACH	PROBLEMS SOLVED
2010 2000	Programming in the world - Software architecture	Mega programs
1990 1980	Programming in the large - Object oriented design - CASE tools - Libraries	Large, complex, distributed
1970 1960	Programming in-the-small - Information hiding	Data intensive, business applications
1950	Programming any-which-way	Simple, algorithmic

Structure Concept in Software

- 1960s – Structured Programming
  - Adopted into programming lanaguages because it's 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 a common practice
  - Specialization of concepts

What is Software Architecture?

- Webster's definition of Architecture
  - The art or science of building:
  - The art or practice of **designing and building structures** and especially habitable ones
  - **A unifying or coherent form or structure**
  - Architectural **product** or work
  - A **method** or **style** of building
  - The **manner** in which computer components/systems are organized and integrated.

What is Software Architecture?

- Architectures are everywhere…
- When you take any artifact, you can say we have an architecture of the artifact. Eg.
  - Building Architecture
  - Car Architecture
  - Computer Architecture
  - Plane Architecture
  - Bridge Architecture
  - Etc..

Discussion…

- What is software architecture according to you?

Software Architecture – Booch 1991

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

- In other words, it is a **high level structure** of software system directed by strategic and tactical design decisions during development.

Software Architecture – Perry and Wolf, 1992

- We distinguish three different classes of architectural elements:
  - Processing elements
    - Those components that supply the transformation on the data elements
  - Data elements and
    - Those that contain the information that is used and transformed
  - Connection elements
    - The glue that holds the different pieces of the architecture together

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.

Software Architecture – IEEE 1471-2000

- Software architecture is the fundamental organization of a system, embodied in its components,
- Their relationships to each other and the environment,
- And the principles governing its design and evolution.

Software Architecture – Bass et al. 2003

- Software architecture of a program or computing system is the **structure or structures of the system**, which comprise software components, the externally visible properties of those components, and the relationships among them.

Summary of Definitions

- Definitions of software architecture has evolved together with technical developments.

- Different definitions but a common agreement on:
  - Architecture includes gross level structure
  - Including components
  - And connections among these components.

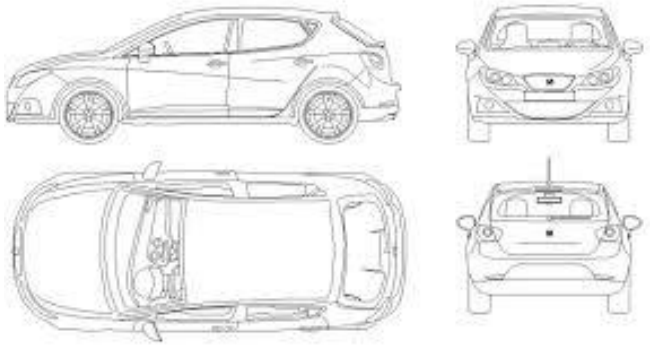
Rationale for Software Architecture

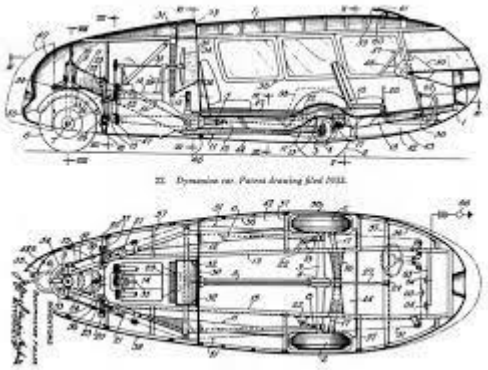
- Discuss
- What is the rationale for software architecture?

- Or, why do we need software architecture?

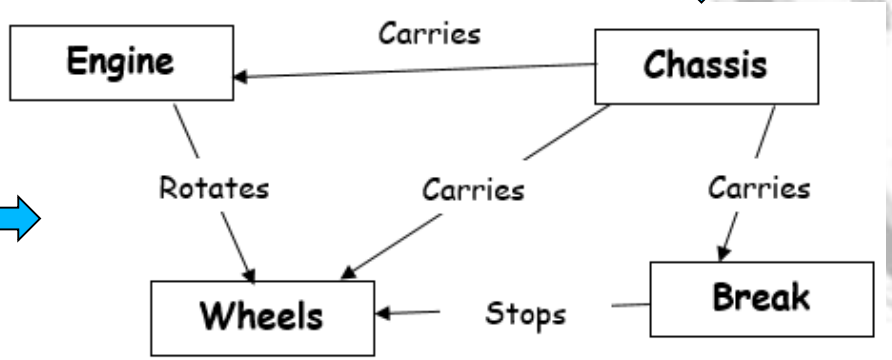
Rationale for Software Architecture

- 1. Abstraction Specification
  - Abstraction
    - Focus only on the relevant properties of the problem
    - Ignore details





Abstraction Level



1. Abstract Specification

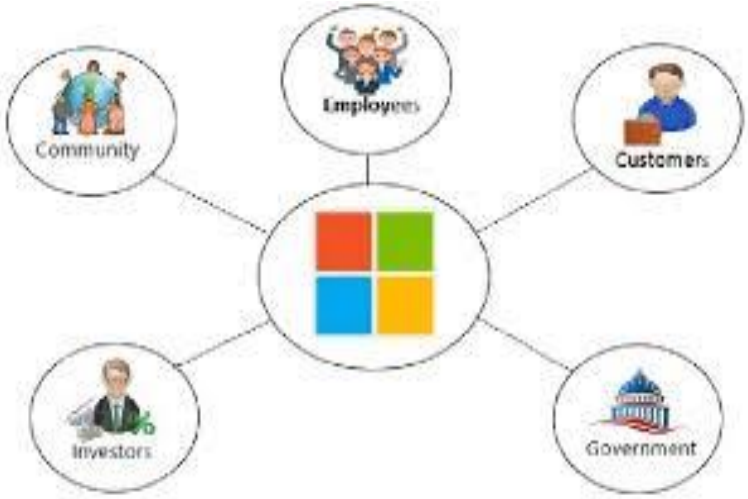
- Architecture represents a **high level abstract specification**.
- Abstraction helps to cope with **complexity**
- Abstraction **improves understanding** of the software system

Rationale for Software Architecture

- 2. Stakeholder Communication
  - Stakeholder is any person who has interest in the architecture
  - Could comprise of:
    - End users
    - Managers
    - Developers
    - Maintainers
    - Analysts
    - Designer
    - Etc..

2. Stakeholder Communication

- Software architecture provides a **common medium for communication** among stakeholders



- This will **improve understanding** of the system among stakeholders

Rationale for Software Architecture

- 3. Coping with Evolution
  - About 80% cost of a 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 in 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**.

Rationale for Software Architecture

- 4. Guides Software Development Process
  - Architecture is explicit
  - Focus on Architectural Components
  - Analyze and design based on architectural components

Rationale for Software Architecture

- 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
    - Basis of integration
    - Basis of test plans, testing
    - Basis of maintenance
    - Incremental deployment

Rationale for Software Architecture

- 6. Large Scale Reuse
  - Software architecture is an abstract specification
  - Representing set of systems
  - Can be reused for system exhibiting similar structure and requirements
  - Can promote software product lines

Rationale for Software Architecture in a Nutshell

1. Improved **understanding** because of a higher level abstract specification.

2. Vehicle for **communication** among different stakeholders because of a common abstract specification.

3. Manifestation of the **earliest design decision**

4. Guides software development **process**

5. Supports organization of **development** project

6. Provides gross level **reuse**

Common Misconceptions

- 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 and Design are the same.
  - All architecture is design, but not all design is architecture (e.g. detailed design).
  - Architecture focuses on significant design decisions.
- Architecture is at a higher abstraction level while design could go down to lower level.

Common Misconceptions

- Architecture is infrastructure of the system.
  - Infrastructure is an important part of architecture.
  - Architecture is more than just infrastructure
  - Infrastructure could be just a view on the architecture.

- Architecture Vs Technology
  - A given technology only serves to implement some dimension of an architecture
    - The network is the architecture
    - The transaction server is the architecture
    - J2EE is the architecture
  - Architecture is more than just a list of products
  - Architecture implementation is shaped by technology, but a robust architecture is not directly bound to technology.

Common Misconceptions

- Architecture Vs Structure.
  - Architecture includes structure, but not every structure is architecture.
  - Architecture is an abstraction of underlying structure.
  - Architecture is more than structure.
    - It also involves behavior, design decisions, constraints…
- Architecture – Views
  - Architecture is flat only in trivial systems in which one architectural view is sufficient
  - However, multiple stakeholders with multiple concerns lead to multiple architectural views.
  - A complex system can usually not be presented using a single architectural view.

Common Misconceptions

- Architecture Vs Art.
  - Current software applications are too complex
  - The "art" or creative part of software architecture is minimal.
  - Architecture design is an explicit rational activity
  - Several architecture design methods exist for this purpose.

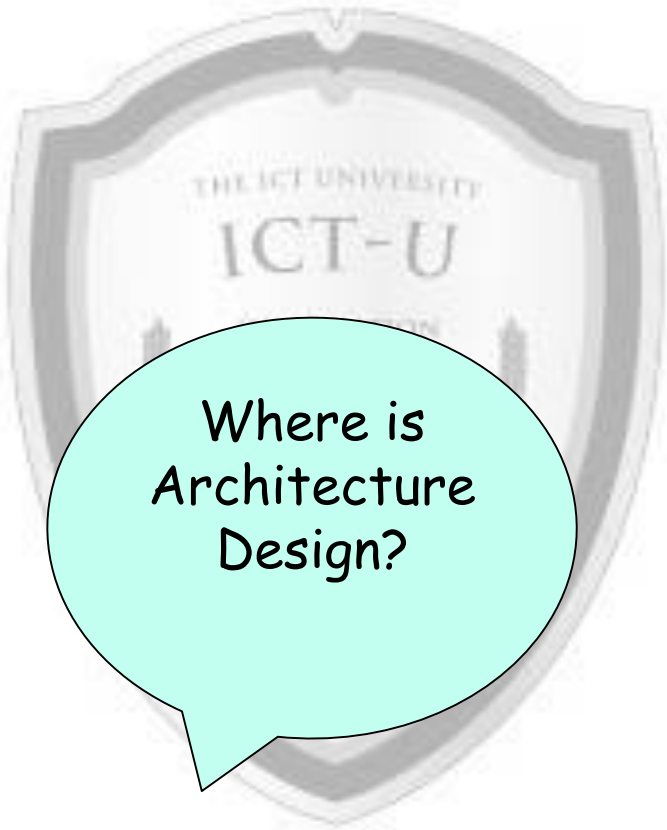
Common Misconceptions in a Nutshell

- Architecture is…
  - Just paper
  - Design
  - Infrastructure
  - Technology
  - Structure
  - Is flat
  - Art
  - Etc..

Software Architecture in the Life Cycle

- Software Engineering Phases

Requirements Analysis	What? (Client)
Analysis	What? (Domain)
Design	How? (Detailed)
Implementation	Do
Testing	Test



Software Architecture – Design Phase

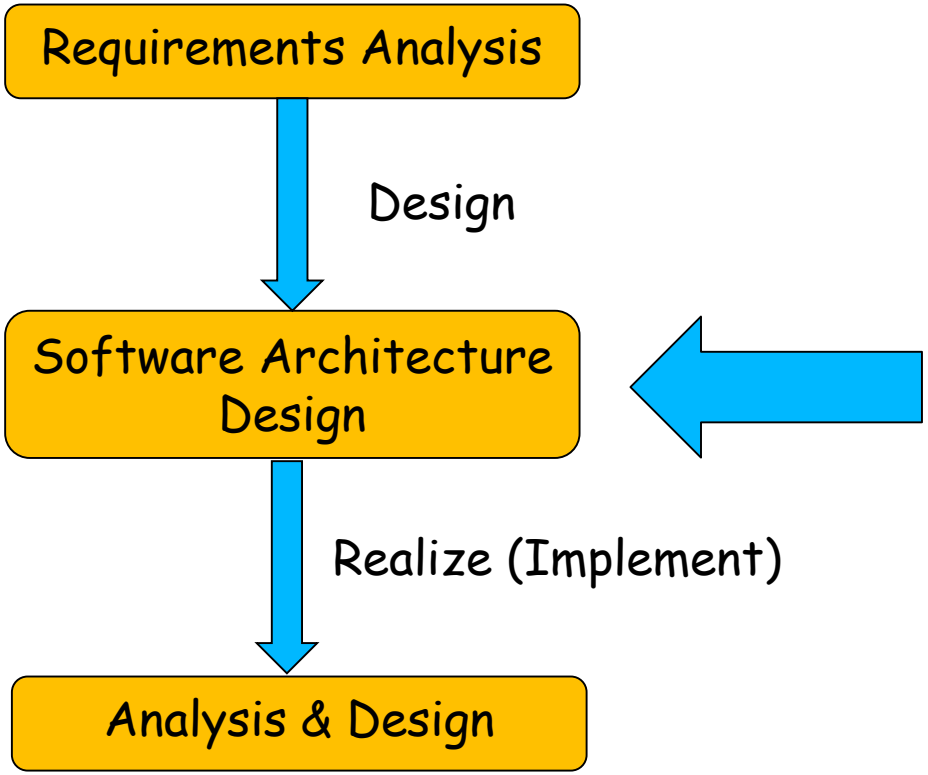
- Software Engineering Phases

Requirements Analysis	What? (Client)
Software Architecture	What? (domain, gross-level)
Analysis	What? (domain, arch components)
Design	How? (Detailed)
Implementation	Do
Testing	Test

Software architecture starts early in the life cycle after Req. Analysis

Design, Evaluate and Realize Architecture

- Software Engineering Phases



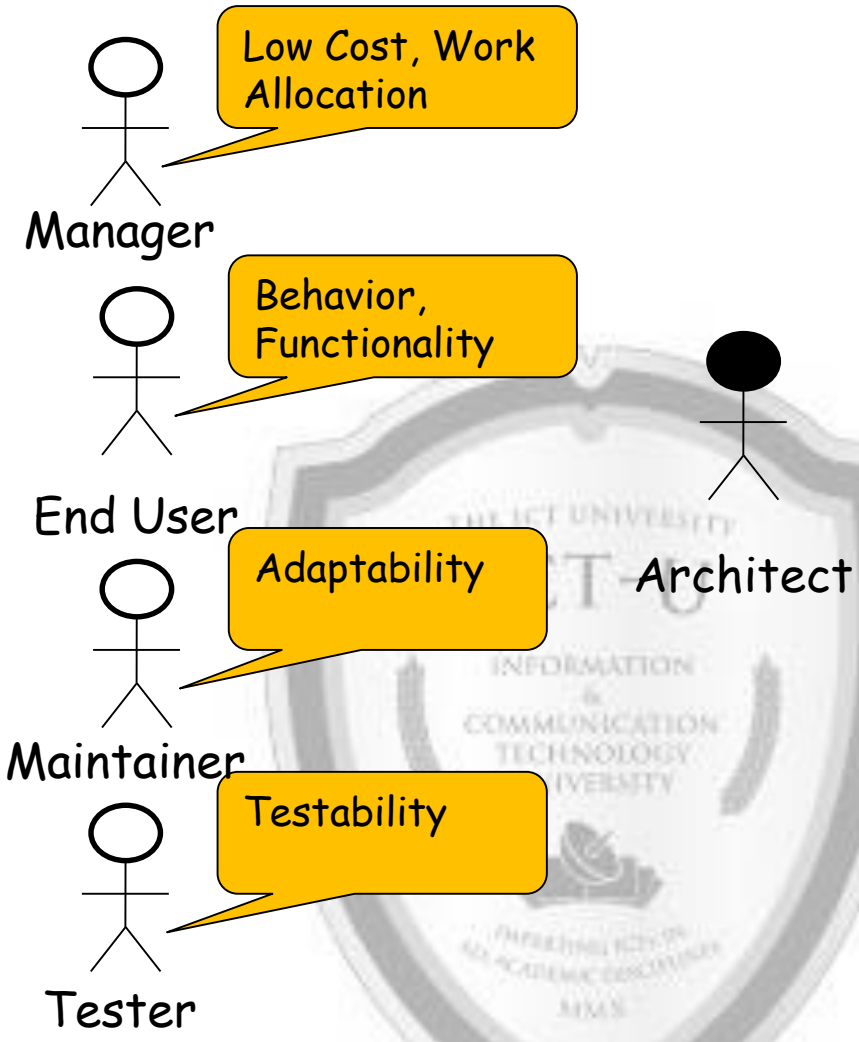
Software architecture starts early in the life cycle after Req. Analysis

Influences on Software Architecture

- What influences the software architecture?
- Software architecture is not done anyhow.
- It is bounded or shaped by some forces.

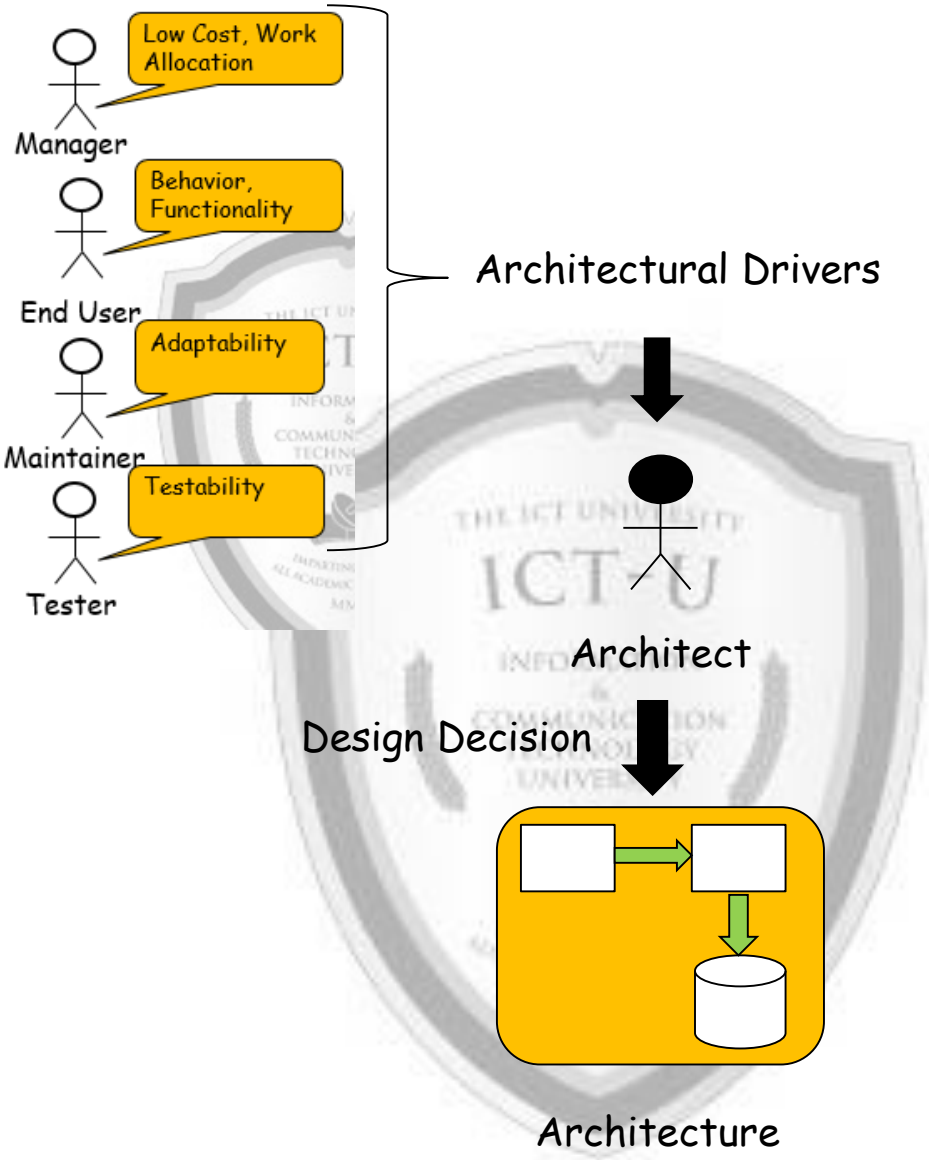
Influences on Architecture - Stakeholders

- Different stakeholders who have interest in the architecture.
- Stakeholders have different concerns
- Some of the concerns might by contra dictionary.
- The architect must be able to identify and balance the stakeholders concerns
- These concerns define the architectural drivers



Architectural Drivers

- Architectural drivers are defined by stakeholder concerns and consist of:
  - High-level functional reqs.
  - Technical constraints
  - Business constraints
  - Quality attribute reqs.
- Each of these exerts force on the architect and influences the early design decisions that the architect makes.
- However, the impact of each on the design can be radically different creating tension



Influences on Architecture - Organization

- An architecture is influenced by the structure or nature of the development organization.
  - E.g. if the organization has somany idle programmers skilled in client-server communications, then a client-server architecture might be the approach supported by the management otherwise, it may be rejected.

- Besides the staff skills, the **development schedule** and **budget** can have a direct influence.

- Goals of the organization
  - Use existing architecture for reuse
  - Focus on specific products

Influences on Architecture - Requirements

- The architecture can affect customer requirements for the next system by giving the customer the opportunity to receive an enhanced system.

- The customer may be willing to relax some requirements to benefit from the architecture (reuse!).

Influences on Architecture - Experience

- An architecture is influenced by the **background** and **experience** of the architect.
- From experience, an architect will use best practices and avoid approaches that did not work for earlier projects
- Architectural choices may also come from an architect's **education** and **training**, exposure to successful architectural patterns
- The more or less experience the architect is, has an impact on the software architecture.

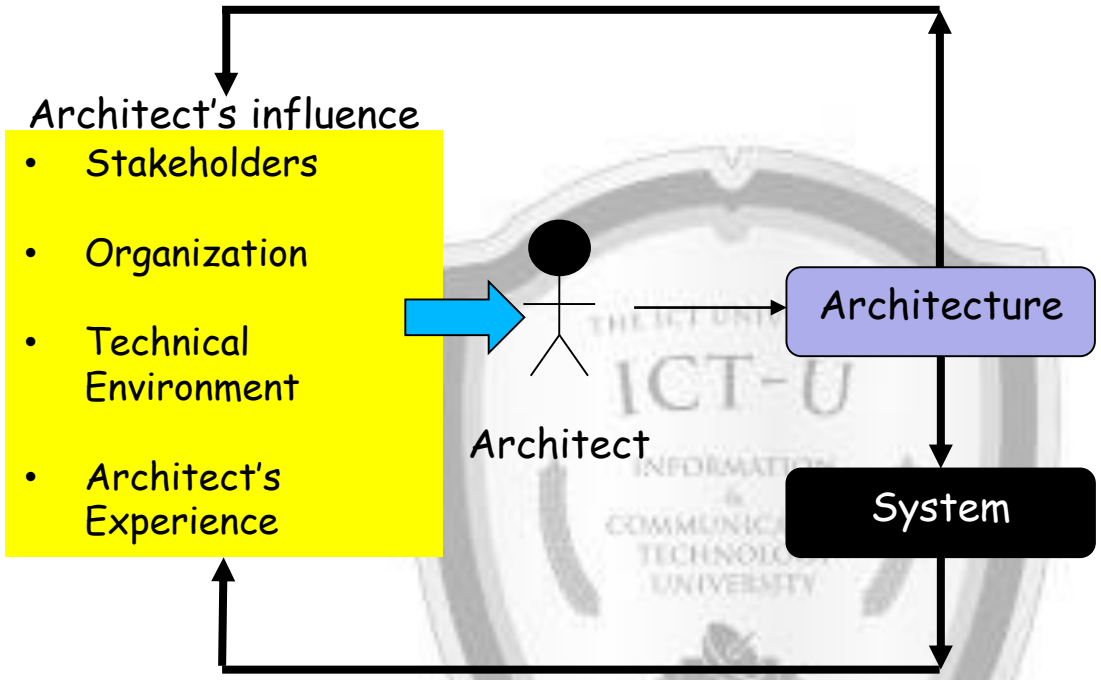
Influences on Architecture - Environment

- Architectures are influenced by technical environment.

- Might include adopted tools, standard industry practices, or software engineering techniques.

Influences on Architecture – Mutual Influences

- Several factors influence the architecture.
- The architecture influences these factors..
- It becomes a cycle of influence..
- Architect needs more than just technical skills. Diplomacy, negotiation and communication skills are essential



Summary

- Software architecture design plays an important role in structuring software
- Several software architecture design definitions which have evolved over the years
- There is now a more mature understanding on the concept of software architecture
- Software architecture represents a high level abstraction which improves understanding the system, supports stakeholder communication, guides subsequent software development process, supports the organization of project, and provides gross level reuse.
- Software architecture is influenced by stakeholder requirements, organization, technical environment and experience of the architect; and vice versa

Q & A