Introduction to Software Architecture

THE UNIVERSITY OF ARIZONA

Something to start with

Conway's Law

- Conway's Law is the theory that organizations will design systems that copy their communication structure.
- Created by computer scientist/programmer Melvin Conway in 1967. Conway's Law states that "Organizations, who design systems, are constrained to produce designs which are copies of the communication structures of these organizations."

THE UNIVERSITY OF ARIZONA

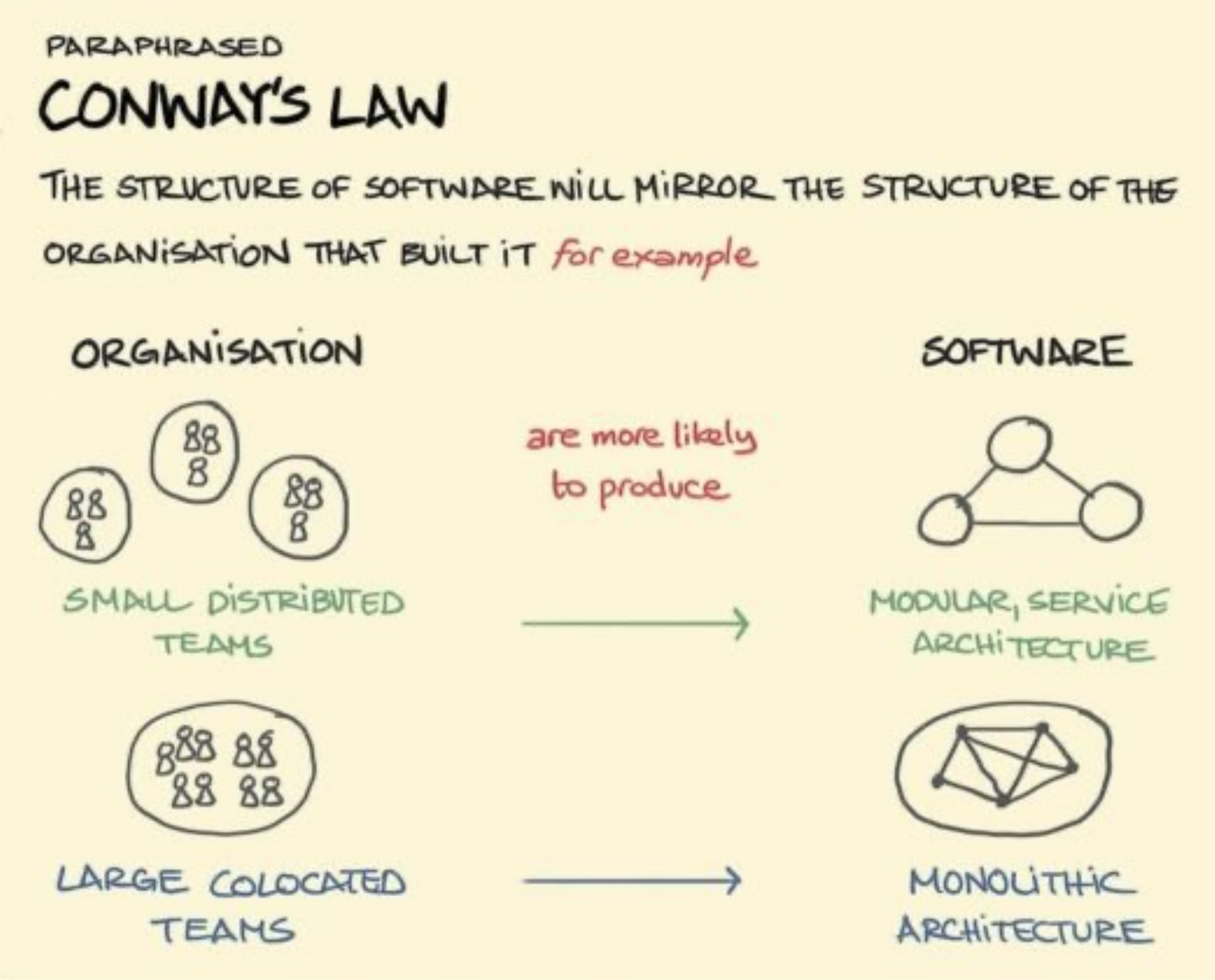
Something to start with

Conway’s Law

- Understanding Conway's Law can help software architects and organizational leaders make more informed decisions about how to structure their teams and software systems.
- By consciously considering the relationship between the organization's structure and software architecture, it is possible to create more effective, maintainable, and scalable software systems.

THE UNIVERSITY OF ARIZONA

Conway's Law



https://en.wikipedia.org/wiki/Conway%27s_law

*https://www.atlassian.com/

What is software architecture?

THE UNIVERSITY OF ARIZONA

* www.uline.com

What is software architecture?

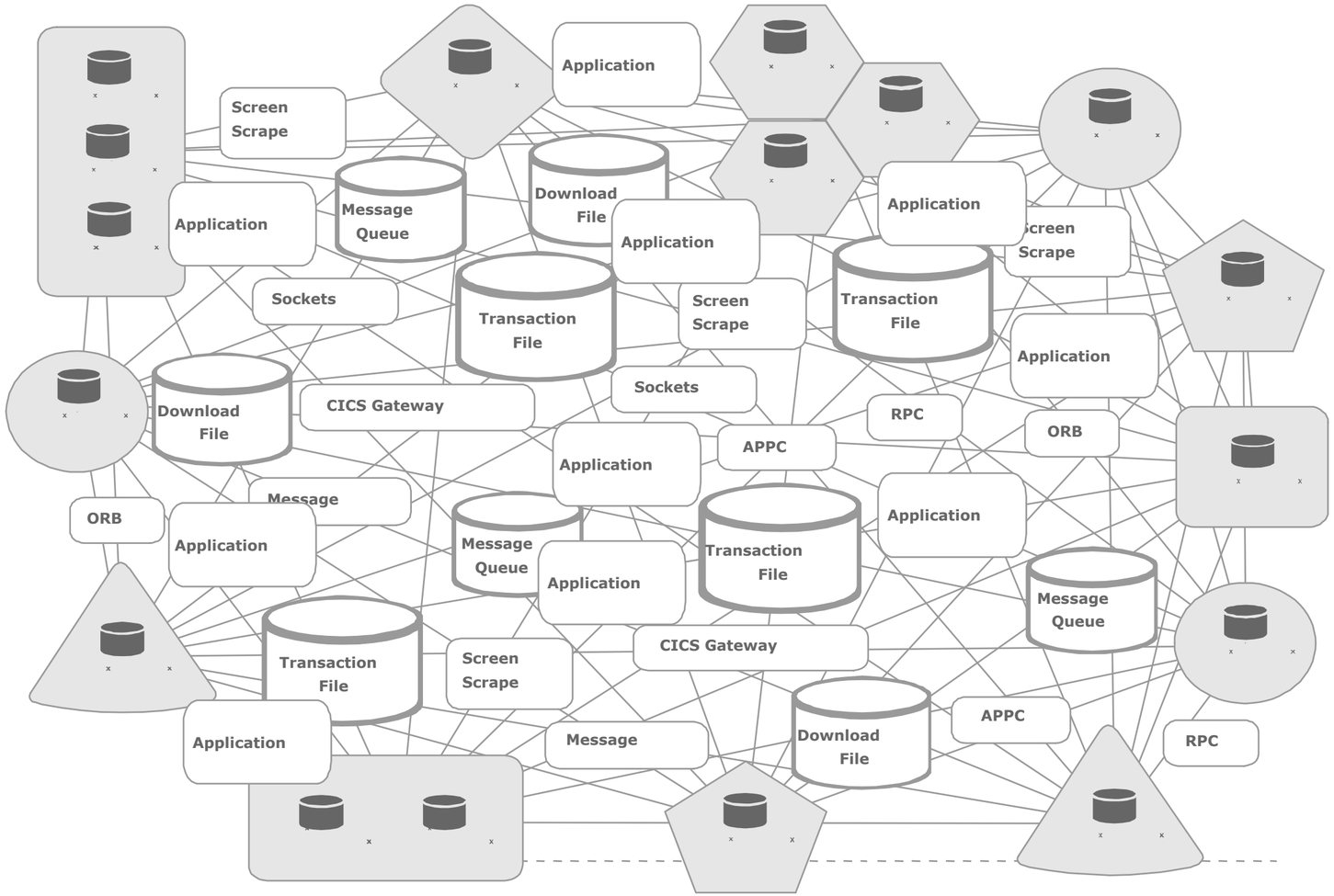
- The high-level structure and organization of a software system

- Classes?
- Methods?
- Parameters?

- Systems of different scales

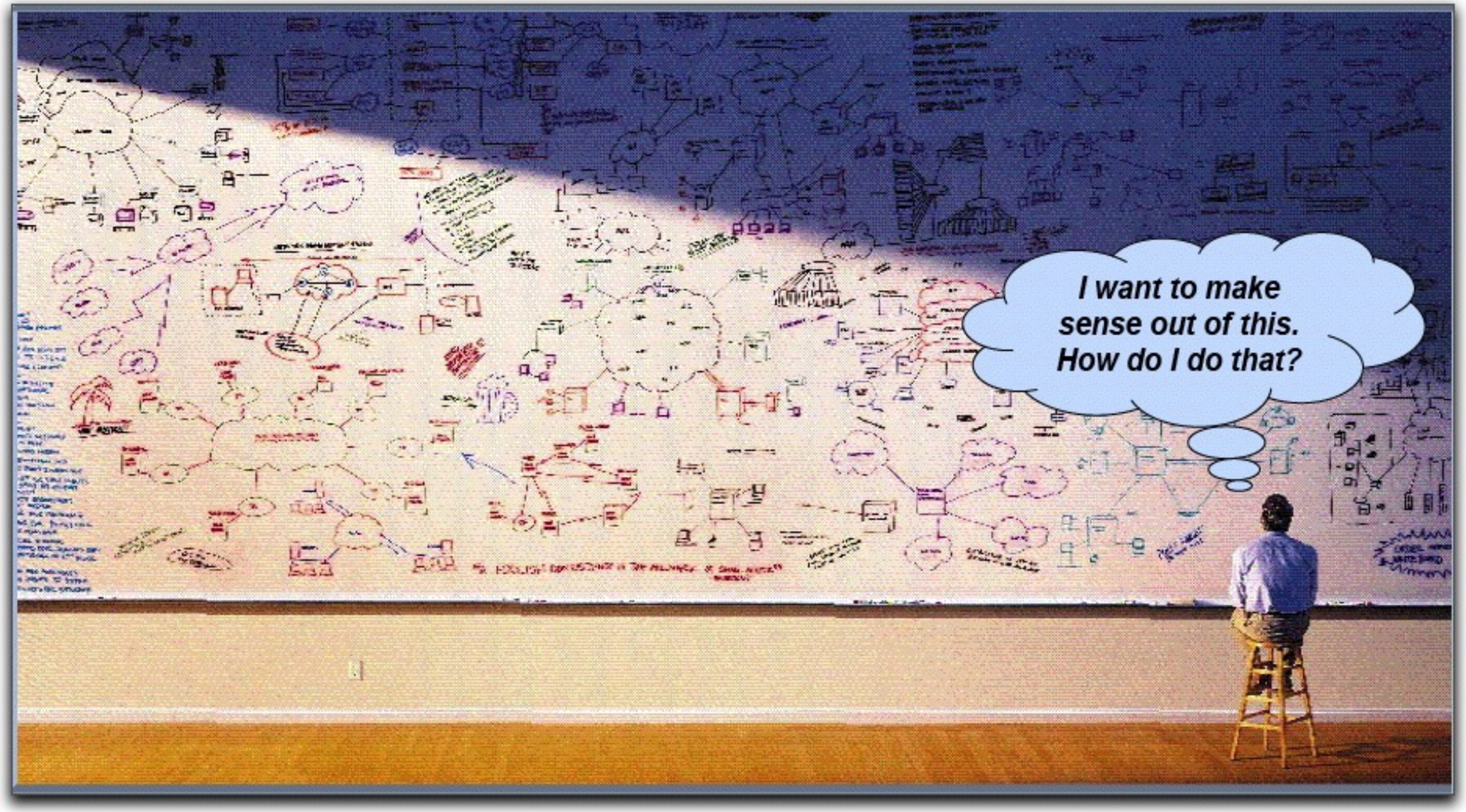
THE UNIVERSITY OF ARIZONA

What is software architecture?



THE UNIVERSITY OF ARIZONA

What is software architecture?



THE UNIVERSITY OF ARIZONA

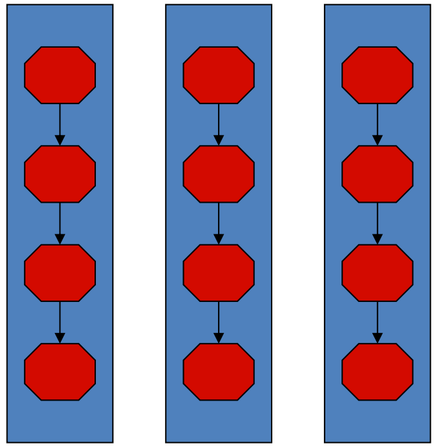
Purpose of Architecture:
To Manage **Interdependencies**

Suppliers	*Logistics*	*Parts*	*Financial*	*Outsourcing*
Manufacturing	*Logistics*	*Engineeering*	*Financial*	*Sourcing*
Management	*Engineeering*	*Financing*	*Warranties*	*Marketing*
Dealers	*Parts*	*Logistics*	*Repair*	*Maintenance*
Customer	*Financing*	*Insurance*	*Taxes*	*Maintenance*

THE UNIVERSITY OF ARIZONA

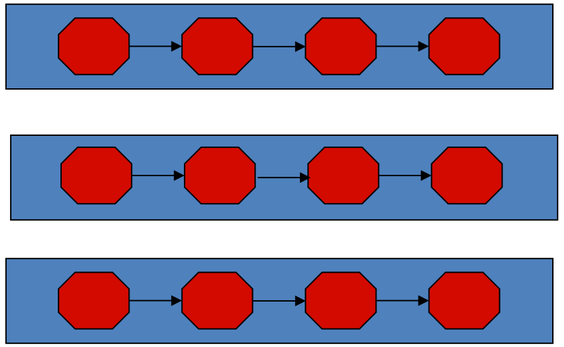
Directions of System Architecture

1960 - 1980



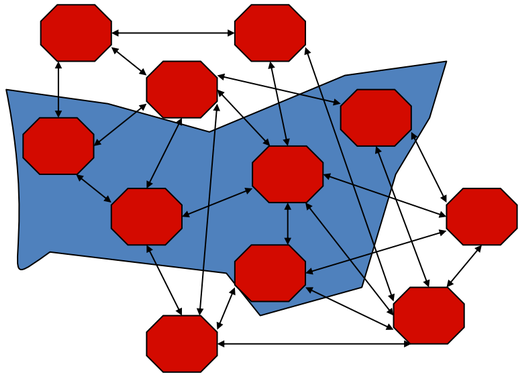
- Organization Focus
- Mainframe Centric
- Internal Use
- Unique Data

1990 - 2000



- Process Focus
- Client Server
- Partial Connectivity
- EDI File Transfer

2010 - 2050

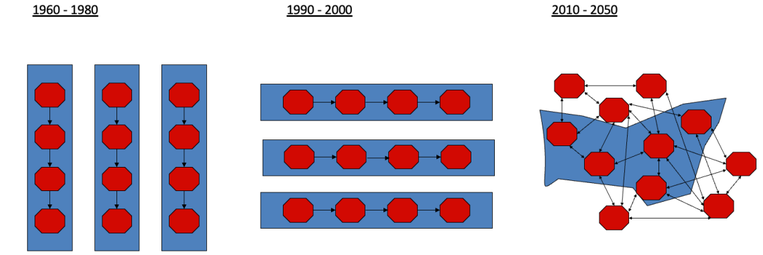


- Distributed Functions
- Data Centric
- Universal Interoperability
- Real-time Connectivity

THE UNIVERSITY OF ARIZONA

Hold on.. Is it software or system architecture?

Directions of System Architecture



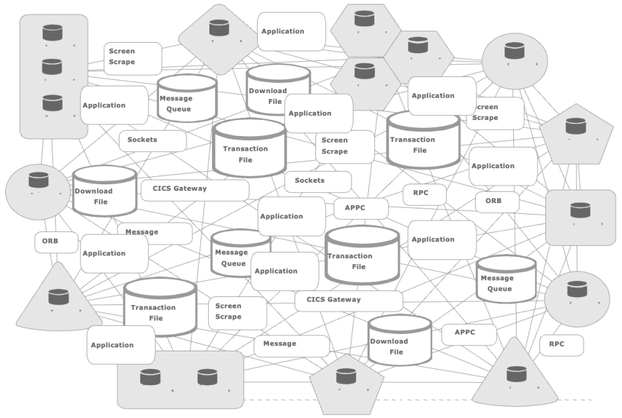
- Organization Focus
- Mainframe Centric
- Internal Use
- Unique Data

- Process Focus
- Client Server
- Partial Connectivity
- EDI File Transfer

- Distributed Functions
- Data Centric
- Universal Interoperability
- Real-time Connectivity

THE UNIVERSITY OF ARIZONA

What is software architecture?



THE UNIVERSITY OF ARIZONA

THE UNIVERSITY OF ARIZONA

Software architecture

- Primarily deals with the structure and organization of the software components and their interactions within a specific software application.
- Concerned with the high-level design of the software,
  - including modules, classes, and their relationships,
  - and how they collectively achieve the software's functionality and quality attributes

System architecture

- Broader in scope and encompasses the entire system,
  - may include not only the software but also hardware components, networks, and other infrastructure elements.
- Focuses on the overall structure and organization of the entire system,
  - including how different subsystems (e.g., software, hardware) work together to achieve a specific set of goals.

THE UNIVERSITY OF ARIZONA

Differences: Abstraction Level

- Software Architecture: operates at a higher level of abstraction, dealing primarily with the software's internal structure, components, and their interactions. **It often abstracts away hardware and infrastructure considerations.**

- System Architecture: operates at a lower level of abstraction, considering the interactions and dependencies between various components, **including hardware**, software, **networks**, and their **physical** or virtual implementations.

THE UNIVERSITY OF ARIZONA

Differences: Components

- Software Architecture: Concerns itself with **software components** like modules, classes, libraries, and their interactions.
- It's more about the software's internal design.

- System Architecture: Encompasses various components, which can include hardware devices (e.g., **servers, routers, sensors**), software applications, middleware, communication protocols, and data storage systems, among others.

THE UNIVERSITY OF ARIZONA

Differences: Stakeholders

- Software Architecture: Primarily addresses the concerns of software **developers**, **architects**, and the **development team**. It aims to provide a blueprint for building the software.

- System Architecture: Addresses the concerns of a broader set of stakeholders, including software and **hardware engineers**, **network administrators, system operators**, and business decision-makers, as it deals with the entire system.

THE UNIVERSITY OF ARIZONA

Differences: Goals

- Software Architecture: Focuses on ensuring that the software system meets its functional and non-functional requirements (e.g., performance, scalability, maintainability) while maintaining a clean and modular codebase.

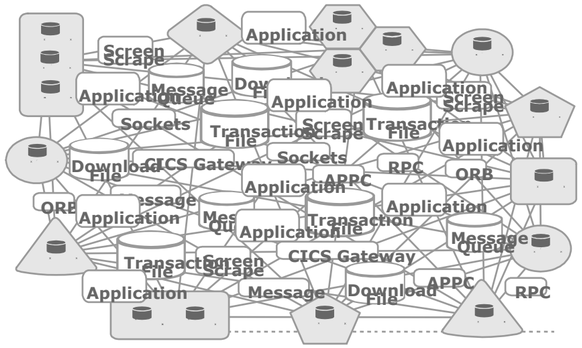
- System Architecture: Aims to achieve the overall goals of the system, which may include optimizing the utilization of **hardware resources**, ensuring high availability and fault tolerance, and meeting business objectives.

Differences: Summary

- In summary, software architecture is a subset of system architecture, and the two are interconnected.
- System architecture considers the bigger picture, encompassing software as one of its components, while software architecture specifically deals with the design and organization of the software portion of the system.

THE UNIVERSITY OF ARIZONA

Since we do the difference..



Architecture

- System decision
- Hard to change
- Scope system

Design

- Module decision
- Moderate/easy to change
- Scope module

THE UNIVERSITY OF ARIZONA

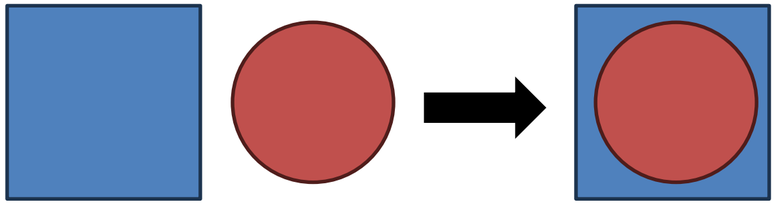
Our focus

- Software Architecture
  - Primarily deals with the structure and organization of the software components and their interactions within a specific software application.
  - Concerned with the high-level design of the software,
    - including modules, classes, and their relationships,
    - and how they collectively achieve the software's functionality and quality attributes
- It often abstracts away hardware and infrastructure considerations
  - We need to be familiar with operating systems and networking protocols

THE UNIVERSITY OF ARIZONA

Software Architecture (other perspectives)

- It is the blueprint or design framework outlining how is a software application built.
- **Software Architecture** acts as a **bridge** between the **system's requirements** and the final implementation.



THE UNIVERSITY OF ARIZONA

Software Architecture (other perspectives)

- The structure and organization of components + interactions + constraints.
- System quality attributes (QA)
  - i.e., how easy it is to modify, how integrable it is..
- Principles and guidelines that guide their design and interaction

THE UNIVERSITY OF ARIZONA

Key aspects of software architecture

- Quality Attributes - addressing non-functional requirements
  (performance, scalability, maintainability, security, and reliability)
- Styles / Patterns - utilizing well-established best practices/building blocks
- Components - identifying the major parts of the software system
- Relationships - defining how these components interact and communicate
- Documentation - architectural documentation for all roles (diagrams, charts, written description)
- Decisions / Trade-offs - balance competing priorities; trade-offs between system aspects
  (including technology choices, frameworks, and methodologies)
- Evolution and Change – evolve the system, adapt, accommodate changing requirements

Key SA aspects we will look into a detail

- Quality Attributes - addressing non-functional requirements
  (performance, scalability, maintainability, security, and reliability)
- Styles / Patterns - utilizing well-established best practices/building blocks
- Components - identifying the major parts of the software system
- Relationships - defining how these components interact and communicate
- Documentation - architectural documentation for all roles (diagrams, charts, written description)
- Decisions / Trade-offs - balance competing priorities; trade-offs between system aspects
  (including technology choices, frameworks, and methodologies)
- Evolution and Change – evolve the system, adapt, accommodate changing requirements

Benefits

1. An architecture will inhibit or enable a system’s driving quality attributes.
2. The decisions made in an architecture allow you to reason about and manage change as the system evolves.
3. The analysis of an architecture enables early prediction of a system’s qualities.
4. A documented architecture enhances communication among stakeholders.
5. The architecture is a carrier of the earliest, and hence most-fundamental, hardest-to change design decisions.
6. An architecture defines a set of constraints on subsequent implementation.
7. The architecture dictates the structure of an organization, or vice versa.

Benefits

8. An architecture can provide the basis for incremental development.

9. An architecture is the key artifact that allows the architect and the project manager to reason about cost & schedule.

10. An architecture can be created as a transferable, reusable model that forms the heart of a product line.

11. Architecture-based development focuses on the assembly of components rather than simply on their creation.

12. By restricting design alternatives, architecture productively channels the creativity of developers, reducing design and system complexity.

13. An architecture can be the foundation for training a new team member.

Summary

- Conway’s law
- Differentiation between system and software
- A rough idea of software architecture

THE UNIVERSITY OF ARIZONA