605.601 Software Analysis and Design
Fall 2020

Module 06A: Software Architecture

Dr. Tushar K. Hazra

tkhazra@gmail.com

(443)540-2230

Software Architecture I Overview

Objectives:

Here is what you should be able to do upon the completion of this module:

- Articulate and describe the role of software architecture in the software engineering process
- Describe and critique the difference in software architectural styles
- Articulate software architecture documentation methods.
- Recapitulate your knowledge of basic software design and architecture

Software Architecture I Overview

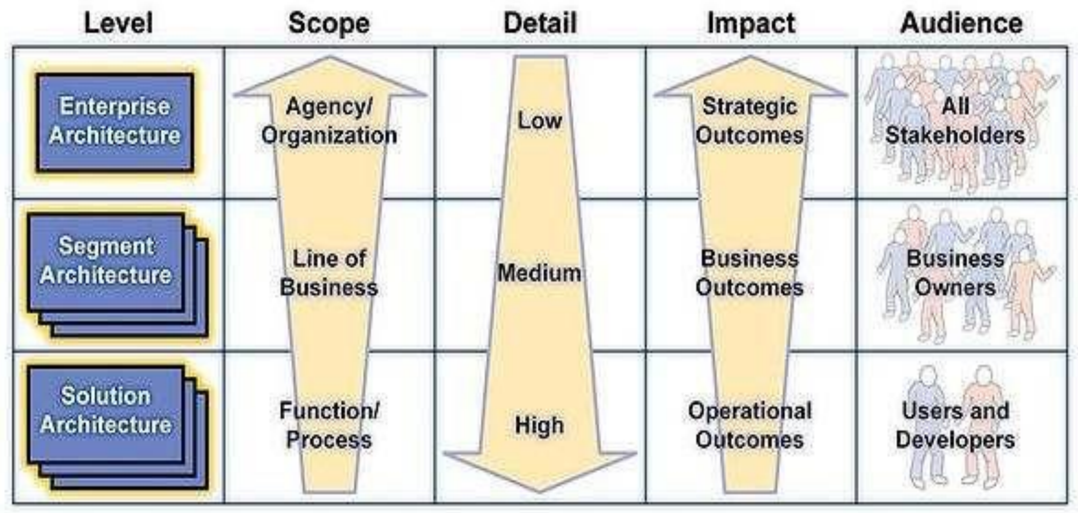
- Software Architecture Definitions
  - Perry and Wolf, 1992
    - A set of architectural (or design) elements that have a particular form
  - Boehm et al., 1995
    - A software system architecture comprises
      - A collection of software and system components, connections, and constraints
  - Clements et al., 1997
    - – The 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

Software Architecture I Overview

- Software Architecture Defined
  - The conceptual structure and overall logical organization of a computer or computer-based system from the point of view of its use

Software Architecture I Overview

- Enterprise Architecture
  - An enterprise architecture has a stronger link to the business in that it focuses on the attainment of the business objectives and is concerned with items such as business agility and organizational efficiency.
  - Software architecture, hardware architecture, organizational architecture, and information architecture, are all subsets of the overall system architecture



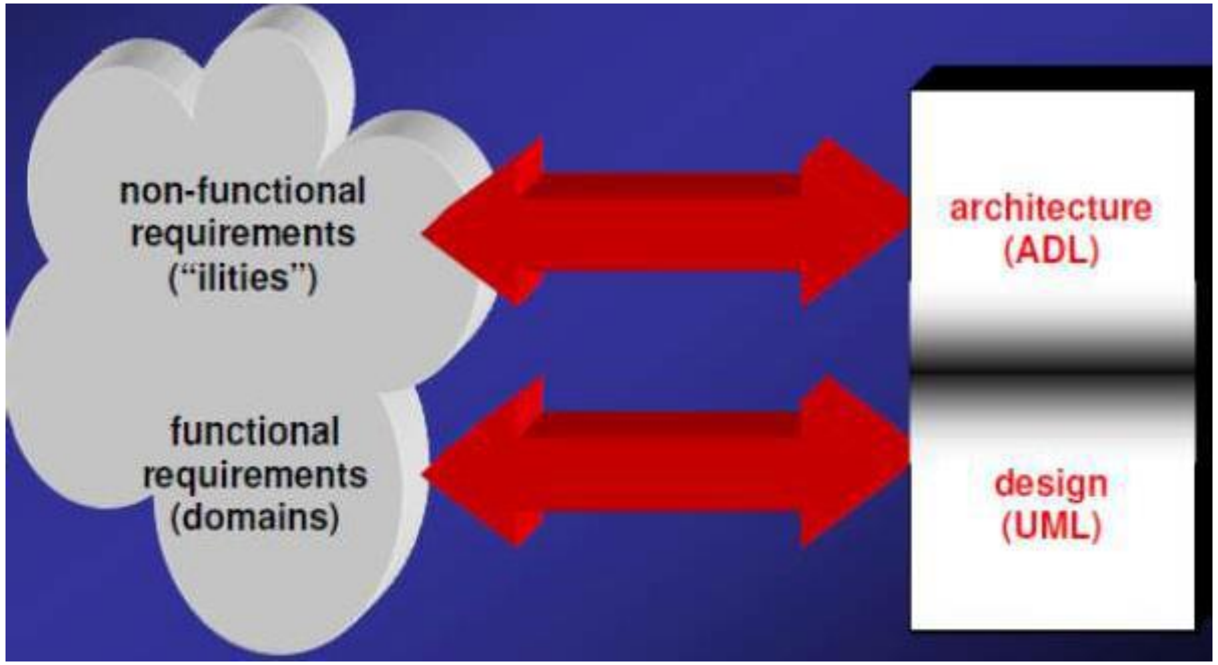
Level	Scope	Detail	Impact	Audience
Enterprise Architecture	Agency/ Organization	Low	Strategic Outcomes	All Stakeholders
Segment Architecture	Line of Business	Medium	Business Outcomes	Business Owners
Solution Architecture	Function/ Process	High	Operational Outcomes	Users and Developers

Software Architecture I Overview

- What Software Architecture is all about
  - Selection of structural elements and their interfaces
  - Behavior
  - Decomposition
  - Architectural styles

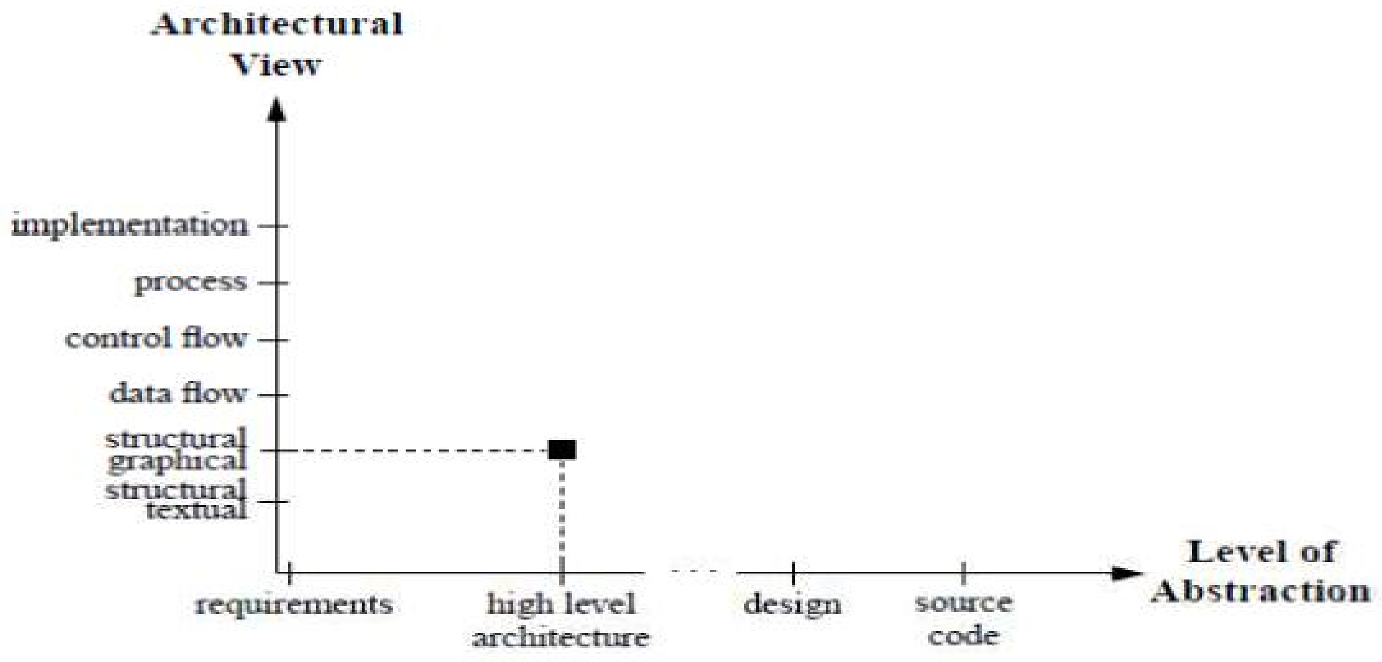
Software Architecture I Overview

- Architecture vs. Design
  - Architecture: where non-functional decisions are cast, and functional requirements are partitioned
  - Design: where functional requirements are accomplished


non-functional requirements ("ilities")
functional requirements (domains)
architecture (ADL)
design (UML)

Software Architecture I Overview

- Why Software Architecture
  - Analyze the effectiveness of the design in meeting its stated requirements,
  - Consider architectural alternatives at a stage when making design changes is still relatively easy, and
  - Reduce the risks associated with the construction of the software.



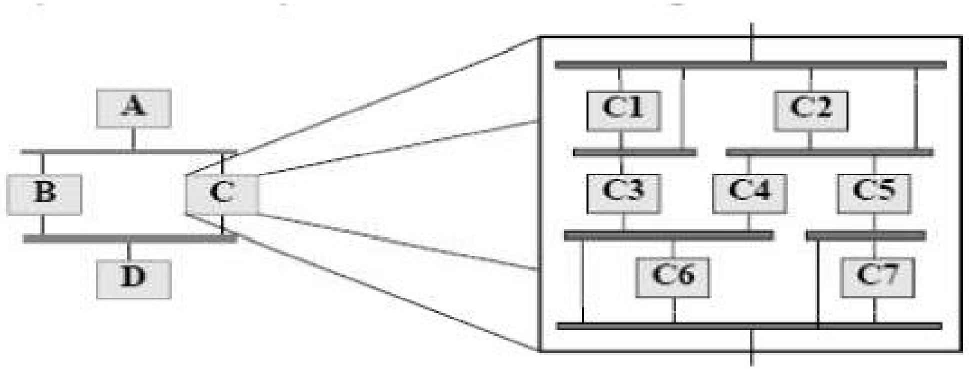
Software Architecture Focus and Scope

Focus and Scope of Software Architecture

- Performance
- Compatibility with legacy software
- Planning for reuse
- Current and future
- Safety, security, fault tolerance
- Evolvability
- Modifications to the structure/functionality

Software Architecture I Overview

- Conceptually Speaking
  - Software architecture should represent a high-level view of the system revealing the structure, but hiding all implementation details.
  - Architecture should realize all the use case scenarios.
  - It should present other systemic views to all the stake holders of the software.
  - Components
  - Configurations



Architecture "constitutes a relatively small, intellectually graspable model of how the system is structured and how its components work together" [BAS03].

Course Module 06A: Software Architecture

- Why is Architecture Important?
  - Representations of software architecture are an enabler for communication between all parties (stakeholders) interested in the development of a computer-based system.
  - The architecture highlights early design decisions that will have a profound impact on all software engineering work that follows and, as important, on the ultimate success of the system as an operational entity.
  - Architecture "constitutes a relatively small, intellectually graspable mode of how the system is structured and how its components work together" (Bass et al., 2003).

Software Architecture

Architectural Patterns

- Architectural patterns are software patterns that describe solutions known to work efficiently, to architectural problems in software engineering.
- It gives description of the elements and relation type together with a set of constraints on how they may be used.
- Defines a specific approach for handling some behavioral characteristics of the system.
- Also known as Architectural Style
- A style/pattern can be thought of as a set of constraints on an architecture, which define a family of architectures that satisfy them.

Software Architecture

Architectural Patterns (continued)

- It expresses a fundamental structural organization schema for a software system, which consists of subsystems, their responsibilities and interrelations.

- In comparison to design patterns, architectural patterns are larger in scale. Even though an architectural pattern conveys an image of a system, it is not an architecture as such.
- An architectural pattern is rather a concept that captures essential elements of a software architecture.

Software Architecture

Sample Patterns

- Concurrency—applications must handle multiple tasks in a manner that simulates parallelism
- operating system process management pattern
- task scheduler pattern
- Persistence—Data persists if it survives past the execution of the process that created it. Two patterns are common:
- a database management system pattern that applies the storage and retrieval capability of a DBMS to the application architecture
- an application level persistence pattern that builds persistence features into the application architecture

Software Architecture

Sample Patterns (continued)

- Distribution— the manner in which systems or components within systems communicate with one another in a distributed environment
- A broker acts as a ‘middle-man’ between the client component and a server component.

Course Module 06A: Software Architecture

- Architectural Patterns
  - Concurrency: Applications must handle multiple tasks in a manner that simulates parallelism
    - .., Operating system process management pattern
    - .., Task scheduler pattern
    - Distribution The manner in which systems or components within systems communicate with one another in a distributed environment
  - .., A broker acts as a “middle-man” between the client component and a server component.

Course Module 06A: Software Architecture

- Architectural Patterns (continued)
  - Persistence: Data persists if it survives past the execution of the process that created it. Two patterns are common:
    - .., A database management system (DBMS) pattern that applies the storage and retrieval capability of a DBMS to the application architecture
    - .., An application-level persistence pattern that builds
  - persistence features into the application architecture

Software Architecture

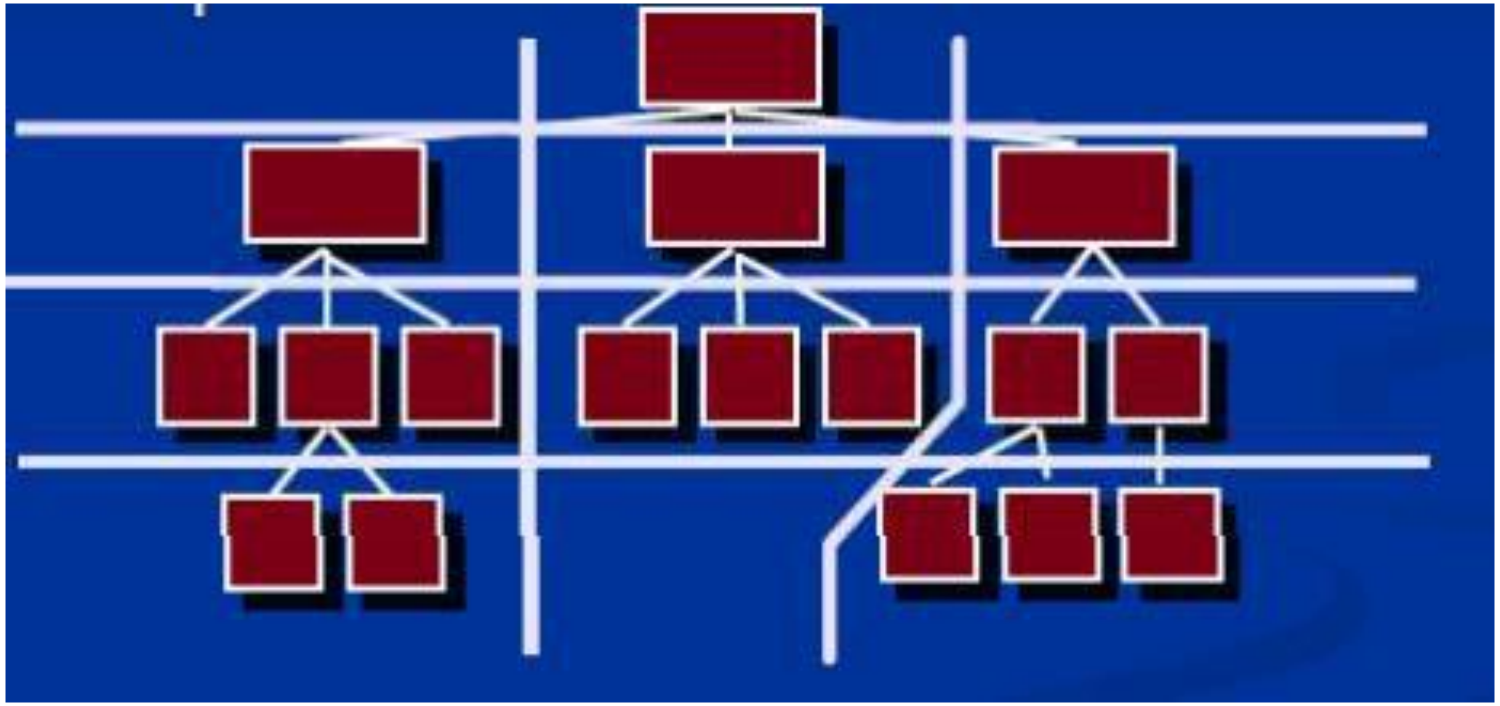
Partitioning – and its Benefits

- Leads to software that is easier to test
- Leads to software that is easier to maintain
- Results in propagation of fewer side effects
- Results in software that is easier to extend

Software Architecture

Partitioning the Architecture

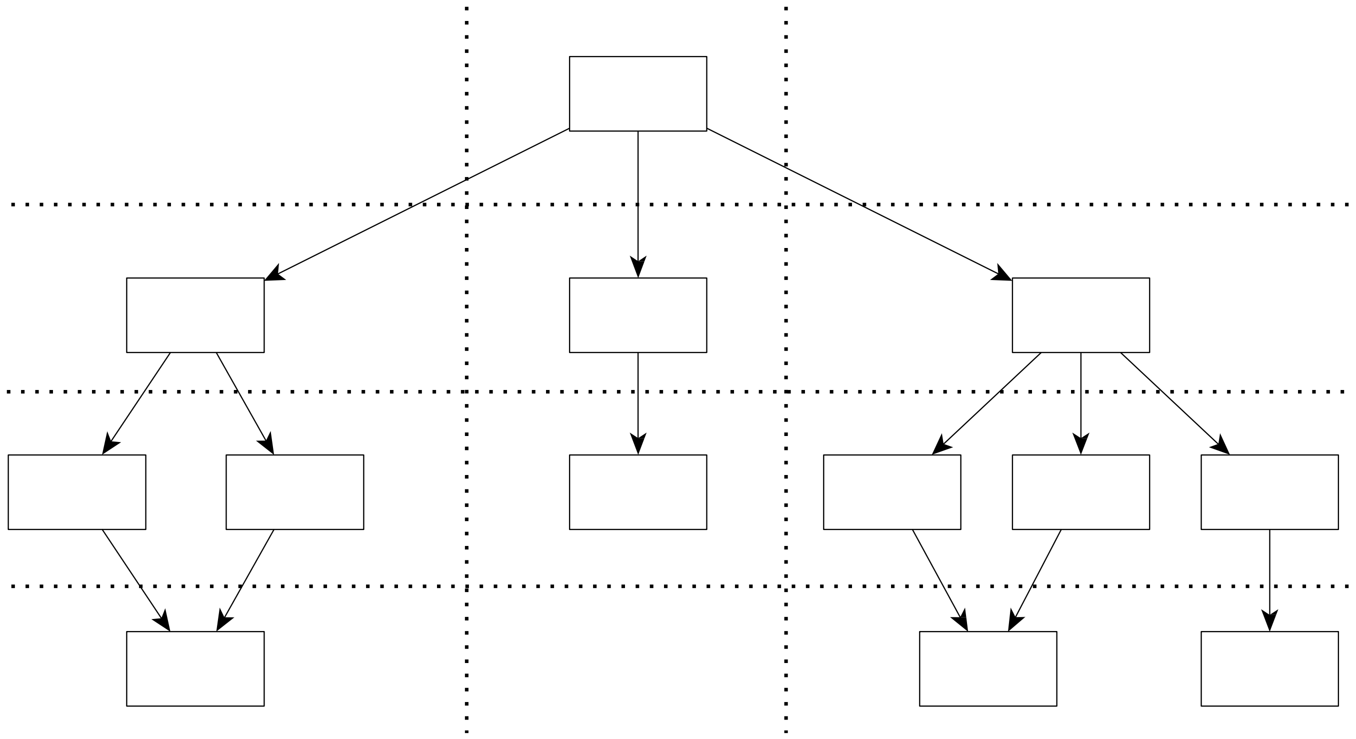
- Horizontal and Vertical Partitioning are required



(from SEPA, 6th ed by Roger G. Pressman)

Course Module 06A: Software Design Foundation

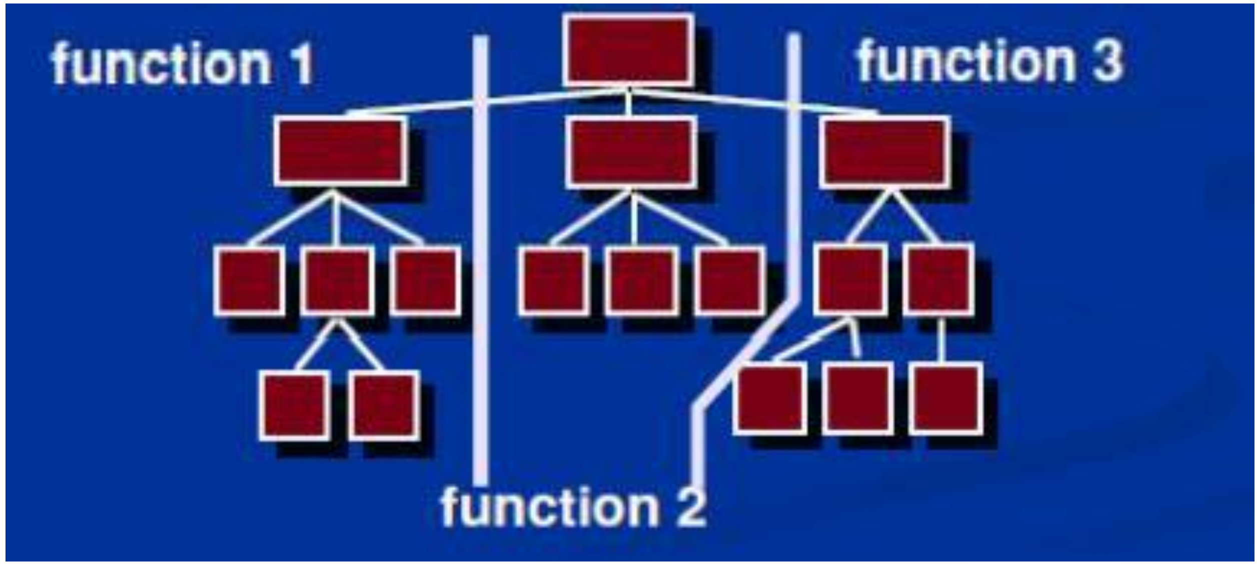
- Partitioning the Architecture
  - “Horizontal” and “vertical” partitioning are required



Software Architecture Styles

Horizontal Partitioning

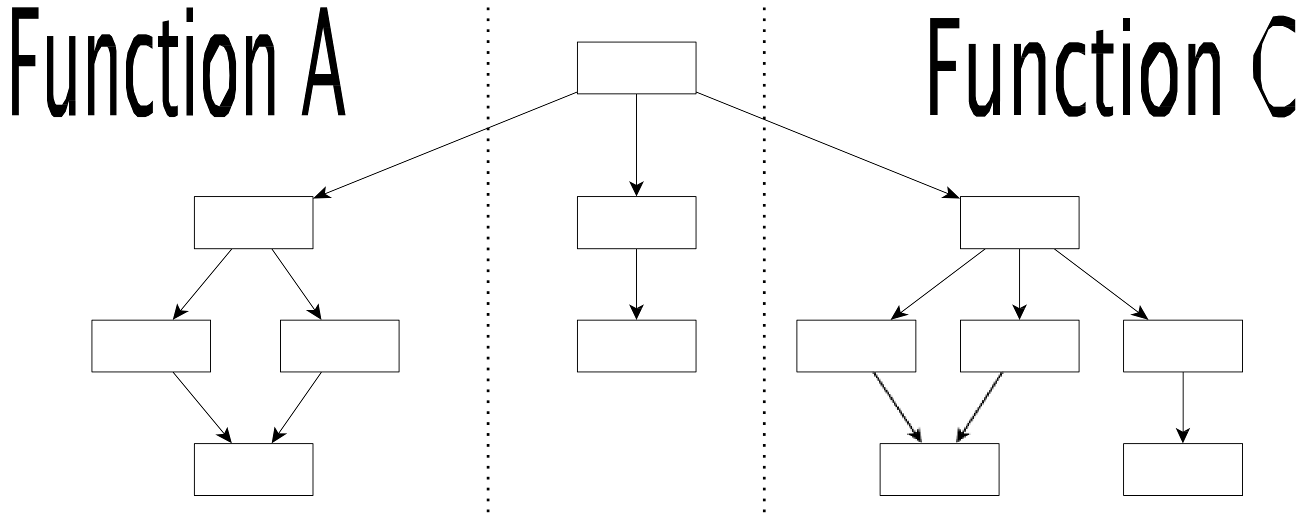
- Define separate branches of the module hierarchy for each major function
- Use control modules to coordinate communication between functions



(from SEPA, 6th ed by Roger G. Pressman)

Course Module 06A: Software Design Foundation

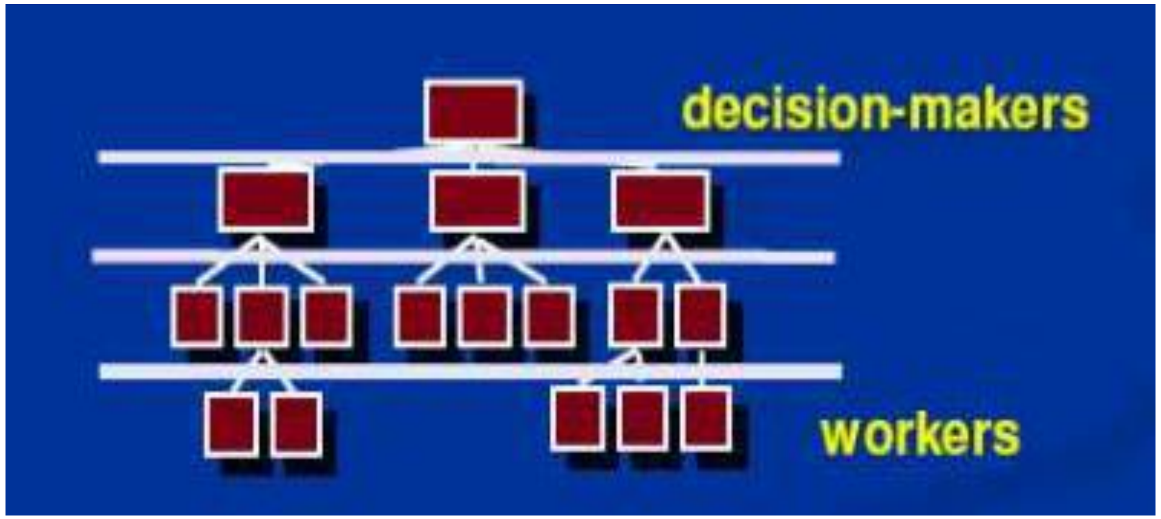
- Horizontal partitioning
- Define separate branches of the module hierarchy for each major function
- Use control modules to coordinate communication between functions



Software Architecture Styles

Vertical Partitioning: Factoring

- Design so that decision making and work are stratified
- Decision making modules should reside at the top of the architecture

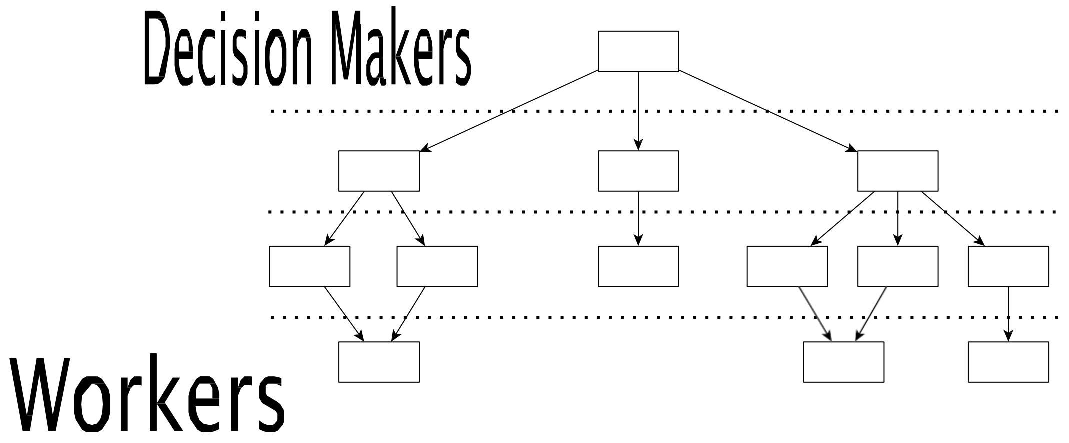


(from SEPA, 6th ed by Roger G. Pressman)

Course Module 06A: Software Design Foundation

Vertical partitioning

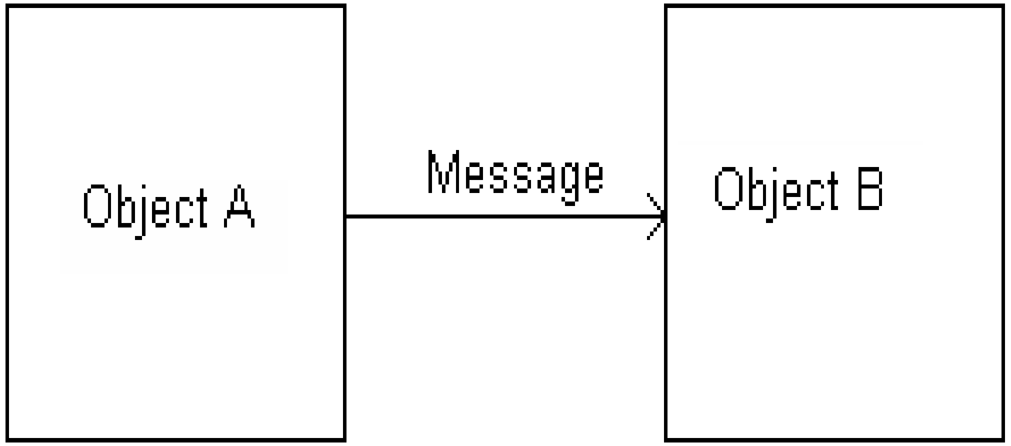
- Design so that decision making and work are stratified
- Decision making modules should reside at the top of the architecture



Software Architecture Styles

Object Oriented Structure

- The components of a system encapsulate data and the operations that must be applied to manipulate the data. Communication and coordination between components is accomplished via message passing

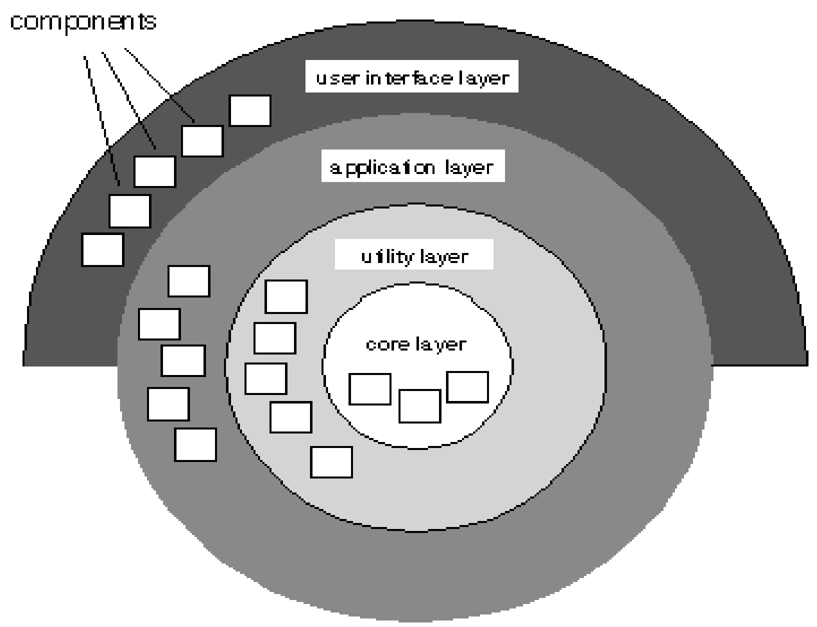


(from SEPA, 6th ed by Roger G. Pressman)

Software Architecture Styles

Layered Architecture

- The layered system is ideal when your system has different levels of functionality.
- – The function you want your system to perform is actually performed at the lowest level, or layer. But you do not want to call the functions at the lowest level directly all the time, because they are complex to use, they have several different implementations, etc.
- At the highest level you just want a few simple functions to do the things you want. Let the lower layers sort the rest out.



(from SEPA, 6th ed by Roger G. Pressman)

Software Architecture Styles

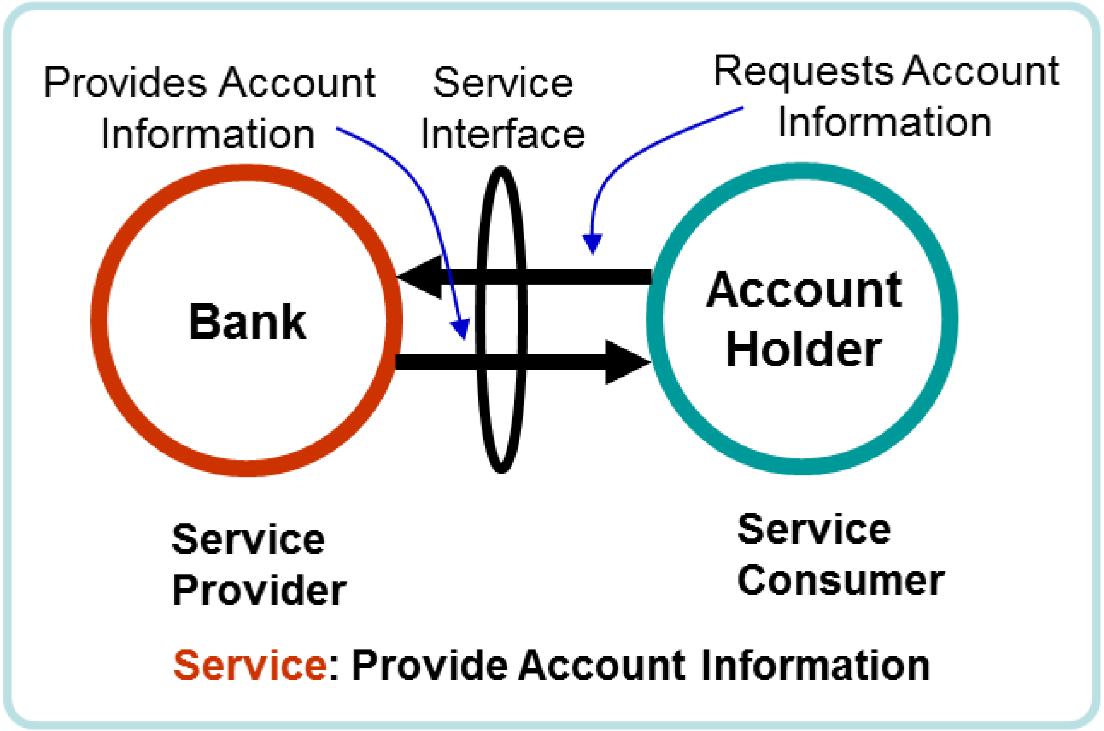
Service Oriented Architecture

- Architectural style that defines an interaction model between three primary parties:
- The **service provider**, publishes a service description and provides the implementation for the service
- A **service consumer**, can either use the uniform resource identifier (URI) for the service description directly or can find the service description in a service registry and bind and invoke the service.
- The **service broker** provides and maintains the service registry
- SOA defines how to integrate widely disparate applications for a Web-based environment and uses multiple implementation platforms. Rather than defining an API, SOA defines the interface in terms of protocols and functionality.

Software Architecture Styles

Service Oriented Architecture – An Example

Service Orientation – is the approach of representing and integrating the functions of a business in terms of a set of accessible services and the outcomes that they produce.



Provides Account Information
Service Interface
Requests Account Information
Bank
Account Holder
Service Provider
Service Consumer
Service: Provide Account Information

Source: Tushar Hazra, The Next Wave of Technologies

Software Architecture Styles

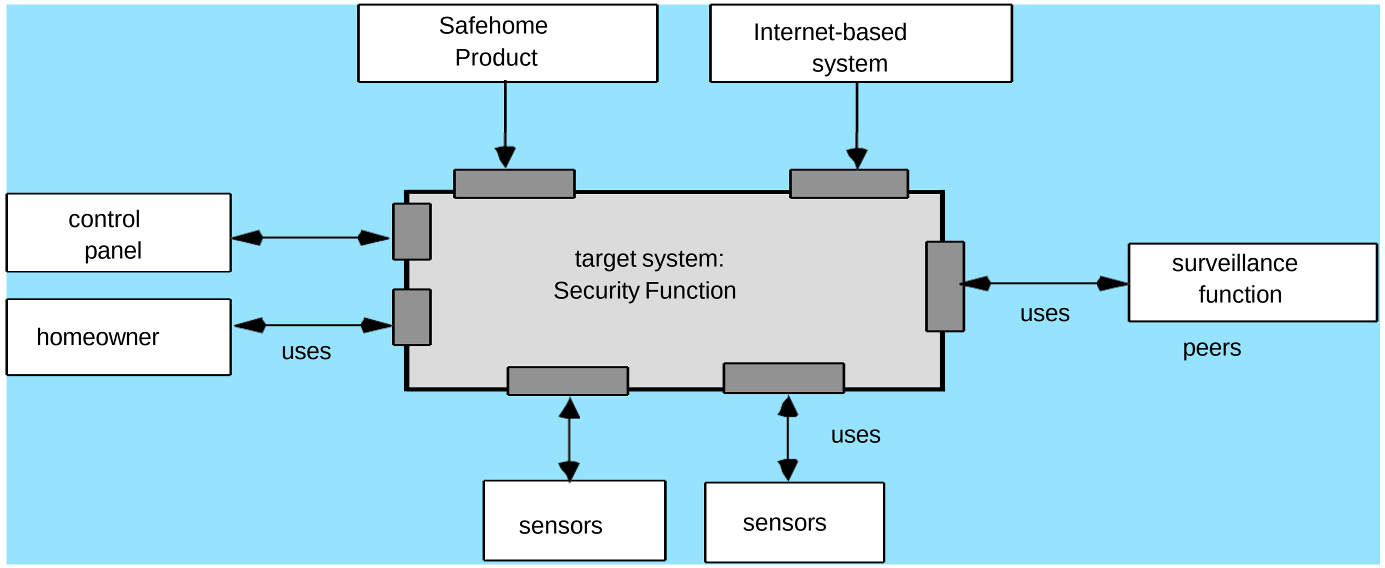
Service Oriented Architecture (continued)

- Architecture packages functionality as a suite of interoperable services that can be used within multiple separate systems from several business domains
- SOA separates functions into distinct units, or services, which are accessible over a network and allow users to combine and reuse them.
- These services and their corresponding consumers communicate with each other by passing data in a well-defined, shared format, or by coordinating an activity between two or more services.

Software Architecture Styles

Architecture Design Steps

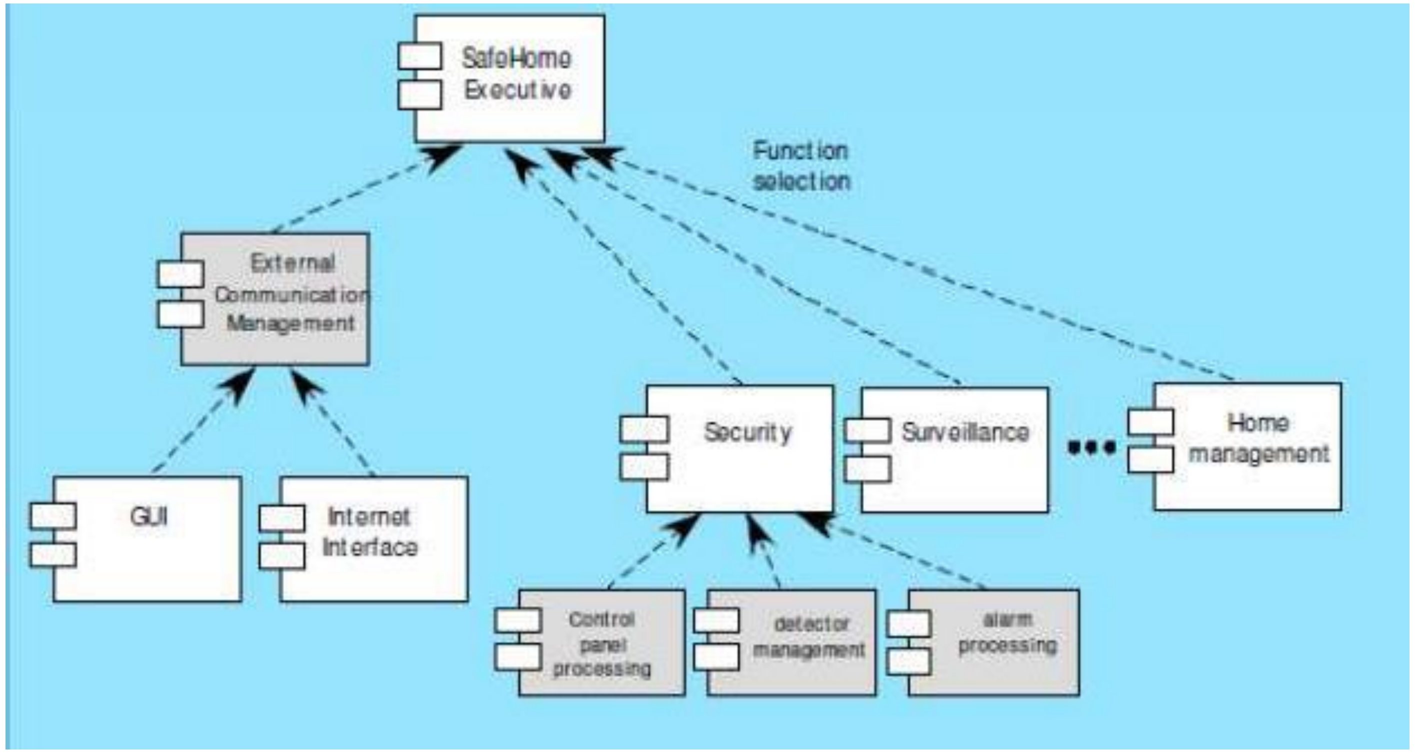
- First, software to be developed must be put into context (external entities with which software interacts)
- Context is generally available from analysis model
- Designer then defines and refines software components that implement the architecture.
- This iterative process is called modular decomposition



(from SEPA, 6th ed by Roger G. Pressman)

Software Architecture Styles

Component Structure



Software Architecture Styles

Step 1 – Meta Architecture

- The meta-Architecture IIs a set of high-level decisions that will strongly influence the structure of the system, but is not itself the structure of the system.
  - The meta-architecture collects together decisions relating to your architecture strategy.
  - It sets direction for your architecture effort, with high-level decisions that will shape the architecture and guide the architects.
  - These include architecture principles, statements of philosophy, metaphors and organizing concepts that will guide system decomposition and design of architectural mechanisms.
- It is intended to shape the architecture, guiding the architecture team, but also guiding technical decisions throughout the life of the architecture.

Software Architecture Styles

Step 2 – System Decomposition

- Software Architecture IIs commonly defined in terms of structural elements and relationships.
- Structural elements are identified and assigned responsibilities that client elements interact with through "contracted" interfaces.
- In creating architectures, we address
- System decomposition into structural elements, architectural components, subsystems, sub-assemblies, parts or "chunks”, paying attention to development productivity, and flexibility or extensibility requirements associated with accommodating future functionality at a reasonable cost of change.

Software Architecture Styles

Key Decomposition Considerations

- A good decomposition satisfies the principle of loose coupling between the pieces, facilitated by clean interfaces, simplifying the problem by dividing it into reasonably independent pieces that can be tackled separately.
- Once broken down into pieces, we need to ask:
- Do we have all the necessary pieces?
- The structure must support the functionality or services required of the system. The dynamic behavior of the system must be taken into account when designing the architecture. We must also have the necessary infrastructure to support these services.

Software Architecture Styles

Key Decomposition Considerations continued

- Do the pieces fit together?
- This is a matter of interface and relationships between the pieces, . Is it a good fit? that is fit that maintains system integrity—also has to do with whether the system, when composed of the pieces, has the right properties.
- Have you addressed crosscutting concerns?
- Are these going to be grouped separately etc ?