Chapter 6

Architectural Design

Architectural Design is concerned with understanding how a system should be organized and designing the overall structure of that system.

* It is the 1st stage in the SW Design Process
* The output of this process is an architectural model that describes how the system ir organized as a set of communicating components.

Agile Processes generally accept that an early stage of the development process should be concerned with establishing overall system architecture. Incremental development of architecture is not usually successful.

You can design SW Architectures at 2 Levels of Abstraction:

* Architecture in the Small
  + Concerned with the architecture of individual programs.
  + Concerned with the way that an individual program is decomposed into components
* Architecture in the Large
  + Concerned with the architecture of complex enterprise systems that include other systems, programs, and program components.

Architecture for SW is important because it affects performance, robustness, distributability, and maintainability.

Individual components implement the Function System Requirements. The Non-Functional Requirements depend on the system architecture. The dominant influence (individual components can also influence) of Non-Functional Requirements is the system architecture.

3 Advantages of Explicitly Designing and Documenting System Architecture

1. Stakeholder Communication
   1. The architecture is a high-level presentation of the system that can be used as a focus for discussion
2. System Analysis
   1. Making the system architecture explicit at an early stage in the development requires some analysis. These decisions have a profound effect on whether or not the system can meet critical requirements such as performance, reliability, etc.
3. Large-scale Reuse
   1. System architecture is often the same for systems with similar components

System Architectures are often modeled using block diagrams

2 Ways an Architectural Model can be used

1. As a way of facilitating discussion about the system design
2. As a way of documenting an architecture that has been designed

Architectural Design Decisions (Section 6.1)

* Is a creative process where you design a system organization that will satisfy the functional and non-functional requirements of a system.
* During this process, architects have to make a number of structural decisions that profoundly affect the system and its development process
  + They have to consider the following questions about the system
    - Is there a generic application architecture that can act as a template for the system that is being designed?
    - How will the system be distributed across a number of cores or processors?
    - What architectural patterns or styles might be used?
    - What will be the fundamental approach used to structure the system?
    - How will the structural components in the system be decomposed into sub-components?
    - What strategy will be used to control the operation of the component in the system?
    - What architectural organization is best for delivering the non-functional requirements of the system?
    - How will the architectural design be evaluated?
    - How should the architecture of the system be documented?
* Although each SW system is unique, system in the same application domain often have similar architectures that reflect the fundamental concepts of the domain.
* Architectures can be based on Architectural Patterns such as Client-Server
* Because of the close relationship between Non-Functional Requirements and Software Architecture. The particular Architecural Style/Pattern and Stucture that you choose for a system should depend on the Non-Functional Requirements:
  + Performance
    - If this is a critical requirement, architecture should localize critical operations within a small number of components on the same computer
  + Security
    - If this is a critical requirement, a layered structure for the architecture should be used
  + Safety
    - If this is a critical requirements, all safety-related operations should be held in one component
  + Availability
    - If this is a critical requirement, the architecture should be designed to include redundant components so that it is possible to replace and update components without stopping the system
  + Maintainability
    - If this is a critical requirement, the architecture should be designed with fine-grained, self-contained components that can easily change

Architectural Views (Section 6.2)

* No matter if the architecture design will be used for a focus for discussion or for a detailed design basis for implementation, there are 2 relevant issus
  + 1) What view or perspectives are useful when designing and documenting a system’s architecture?
  + 2) What notations should be used for describing architectural models?
* There should be 4 fundamental views
  + 1) Logical View
    - Show key abstractions in the system as objects or object classes
  + 2) Process View
    - Shows how at run-time they system is composed of interacting processes
  + 3) Development View
    - Shows how the SW is decomposed for development…shows the breakdown of the SW into components that are implement by a single developer or development team
  + 4) Physical View
    - Shows the system hardware and how software components are distributed across processors in the system.
* Users of Agile Methods suggest that detailed design documentation is mostly unused. It is simply a waste of time and money to develop it.

Architectural Patterns (Section 6.3)

* These are stylized, abstract description of good practice/methods, which has been tried and tested in different systems and environments.
* Layered Architecture (Section 6.3.1)
  + Used to achieve separation and independence.
  + Supports incremental development
  + Info
    - Description
      * Organizes the system into layers with related functionality associated with each layer. A layer provides services to the layer above it so the lowest-level layers represent core services that are likely to be used throughout the system
    - When Used
      * When incorporating new functionality on top of existing systems
      * When development is spread across teams each assigned a layer
      * When there is a need for multi-level security
    - Advantages
      * Can add/remove layers as long the interface is the same
    - Disdavantages
      * Is difficult
      * Layers may need to skip layers in order to interact with other layers
* Repository Achitecture (Section 6.3.2)
  + Shows how a set of interacting components can share data
  + Info
    - Decription
      * All data in a system is managed in a central repository that is accessible to all system components. Components do not interact directly, only through the repository
    - When Used
      * When you have a system in which large volumes of info is gathered and needs to be stored for a long time
      * You it also when the inclusion of data in the repository triggers an action or tool
    - Advantages
      * Components can be independent
      * Changes made to one component can be propogated to all components
      * All data can be managed consistently
    - Disadvantages
      * Repository is a single point of failure
      * Can be inefficient in organizing all communication through the repository
* Client-Server Architecture (Section 6.3.3)
  + Concerned with the static structure of the system and does not show its run-time organization.
  + Organized as a set of services and associated servers, and clients that access and use the services
  + Major Components
    - A set of servers that offer services to other components
    - A set of clients that call of the services offered by servers
    - A network that allows the clients to access these services
    - Info
      * Description
        + The functionality of the sytem is organized into services, with each service delivered from a separate server. Clients are users of these services and access servers to make use of them
      * When Used
        + Used when data in a shared database has to be accessed from a range of locations
      * Advantages
        + Servers can be distributed across a network
        + General functionality can be available to all clients and does not need to be implement by all services
      * Disdadvantages
        + Each service is a single point of failure so susceptible to denial of service attacks or server failure
        + Performance may be unpredictable due to the network
* Pipe and Filter Architecture (Section 6.3.4)
  + Model of Run-Time organization of a system where functional transofmrations process their inputs and produce outputs
  + Data flows from one to another and is transformed as it moves through the sequence
  + Info
    - Description
      * Processing of the data in a system is organized so that each processing component (filter) I sdiscrete and carries one type of data transformation. The data flows (pipe) from one component to another for processing
    - When used
      * Commonly used in data processing applications
    - Advantages
      * Easy to understand and supports transformation reuse
      * Can easily evolve
    - Disadvantages
      * Format for data transfer must be agreed upon
      * Overhead exists because each process must parse it input and unparsed its ouput

Application Architectures (Section 6.4)

* Application Architectures can be used in a number of ways:
  + As a starting point for the architectural design process
  + As a design checklist
  + As a way of organizing the work of the development team
  + As a means of assessing component for reuse
  + As a vocabulary for talking about types of applications
* 2 Types of Application Architecure
  + Transaction processing applications
    - Database centered applications that process user requests for information and update the info in a database
  + Language Processing Systems
    - System in which the user’s intentions are expressed in a formal language. The system processes this language into an internal format and then interprets this internal representation.
      * Ex. Compilers

Transaction Processing Systems (Section 6.4.1)

* Designed to process user requests for information from a database, or requests to update a database.
* Changes will only be record if they finish successfully
  + Ex. Until all steps have been finished by users at ATM like enter card, pin, money wanted, etc. the database won’t record the deposit or withdrawal.
* Can be organized as PIPE AND FILTER ARCHITECTURES

Information System (Section 6.4.2)

* ?? <Is this the same as Transaction Processing System?>
* Can be organized as Client Server Architecture

Language Processing System (Section 6.4.3)

* Translate a natural or artificial language into another representation of that language, and for, programming languages, may also execute the resulting code.
* Can be used with a Pipe and Filter Architecture or Repository Architecture