Software Architecture

SEN-522 Advanced Software System Architecture

(Fall 2020)

Dr Awais Majeed

Department of Software Engineering
Bahria University, Islamabad

Topics Covered

Software Architecture

- As the size and complexity of software systems increases, the design and specification of overall system structure become more significant issue than the choice of algorithms and data structures.
- Structural issues include system organization as a composition of components, protocols for communication, interaction through interfaces, etc.
- This is the ***Software Architecture*** level of design.

What is Software Architecture?

- A natural evolution of design abstractions.
- Involves the description of elements from which system is built, interactions among those elements, patterns that guide their composition, and constraints on the patterns.
- Considers system as a collection of components and their interactions.

Components - Interactions

- **Components** are such things as clients and servers, databases, layers, etc.
- **Interactions** among components can be procedure calls, shared variable access, etc.
- At the architectural level, we also consider **system-level issues** such as *capacity*, *consistency*, *performance*, etc.

Definitions

The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [IEEE standard 1471–2000].

Definitions

The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [IEEE standard 1471–2000].

- Software Architecture defines the observable properties of a software system

Definitions

The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [IEEE standard 1471–2000].

- Software Architecture defines the observable properties of a software system

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

– Len Bass

- **Software architecting** means the process of creating software architectures.

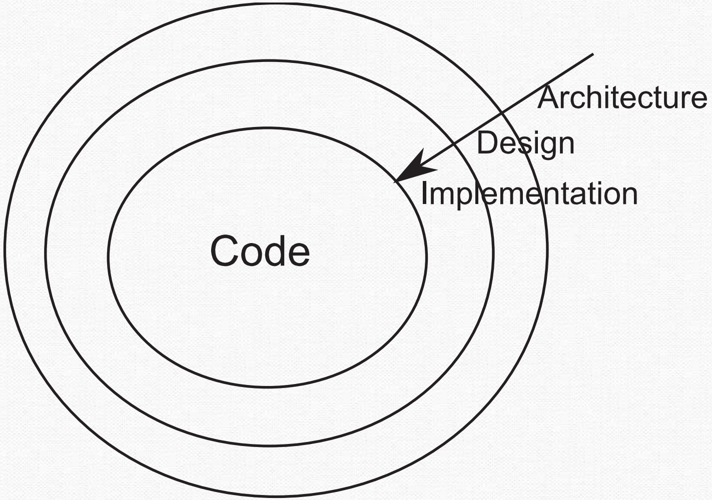
Architecture vs Design I

- “All architecture is design, not all design is architecture”
- Architectural design is outward looking
  Focus on stakeholders, not technology
- Architecture doesn’t describe the complete characteristics of components – Design does.

Architecture vs Design II

- Furthermore, the actual placement of the software architecting process within the software life-cycle further distinguishes it from the classical design phase.
- In essence, the software architecting process has expanded the traditional design phase to include a phase that immediately precedes the traditional high-level design phase and will at times overlap this phase depending on the architectural view.

Architecture vs Design III



Architectural Influences

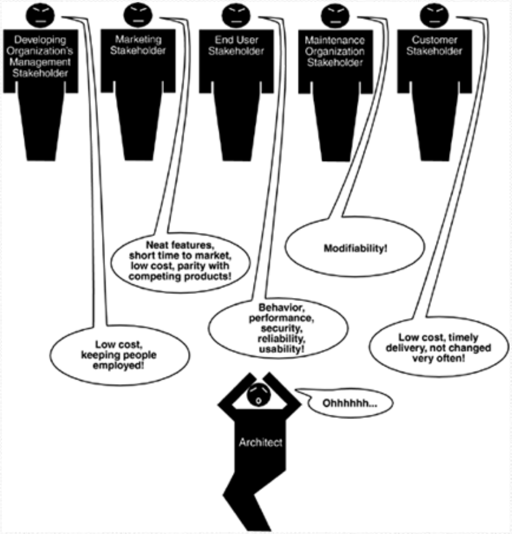
Influences

- System Stakeholders
- Developing organization
- Architects' background and experience
- Technical environment

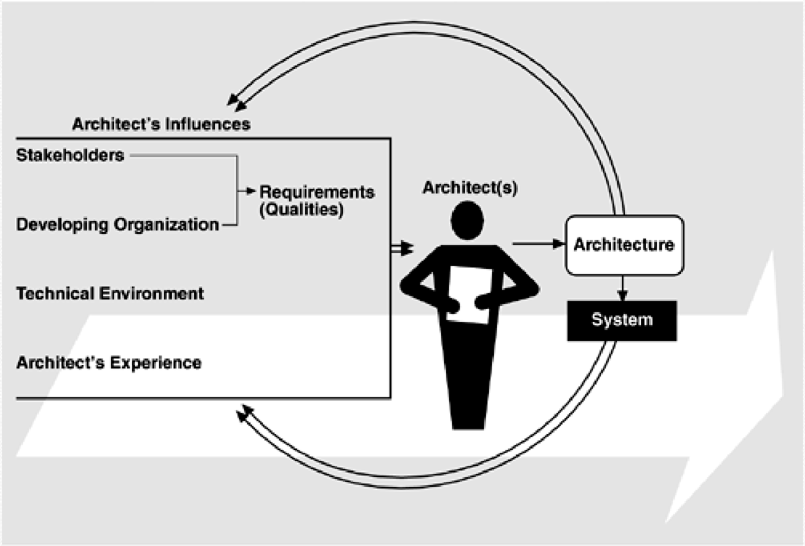
Ramification of influences on architecture outcome/result

- Know your constraints
- Early engagement of stakeholders

Stakeholder influences


Developing Organization's Management Stakeholder
Marketing Stakeholder
End User Stakeholder
Maintenance Organization Stakeholder
Customer Stakeholder
Neat features, short time to market, low cost, parity with competing products!
Modifiability!
Behavior, performance, security, reliability, usability!
Low cost, keeping people employed!
Low cost, timely delivery, not changed very often!
Ohhhhhh...
Architect

ABC Cycle


Architect's Influences
Stakeholders
Requirements (Qualities)
Developing Organization
Technical Environment
Architect's Experience
Architect(s)
Architecture
System

Architecture Business Cycle (ABC)

- Software architecture is a result of technical, business and social influences.
- These are in turn affected by the software architecture itself.

This cycle of influences from the environment to the architecture and back to the environment is called the Architecture Business Cycle (ABC).

ABC Activities I

Creating the business case for the system

- Why we need a new system, what will be its cost? Time to market, integration with existing systems?

Understanding the Requirements

- Various approaches for requirements elicitation i.e., object-oriented approach, finite-state machine models, prototyping etc.
- The desired qualities of a system shape the architectural decisions – architecture defines the trade-offs among requirements

Creating/selecting the architecture

ABC Activities II

Communicating the architecture

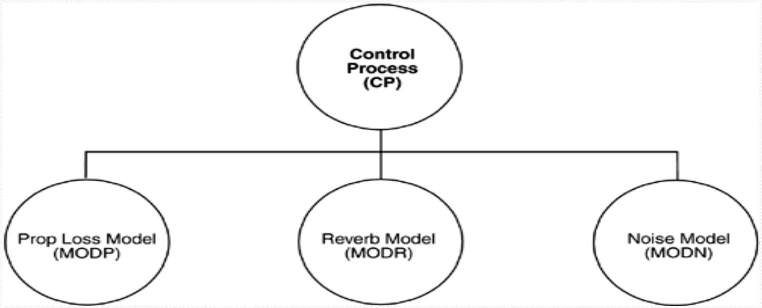
- Inform all stakeholders (i.e., developers, testers, managers, etc.)
- Architecture's documentation should be unambiguous

Analysing or evaluating the architecture

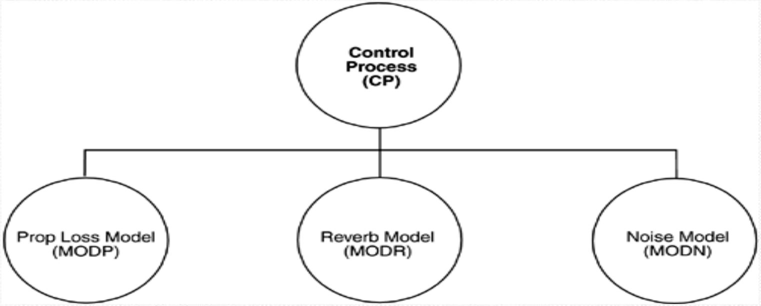
- Evaluate candidate designs
- Architecture maps the stakeholders' requirements/needs

Implementation based on architecture
Ensuring conformance to an architecture

Software Architecture



Software Architecture



1. *What is the nature of the elements?*
2. *What are the responsibilities of the elements?*
3. *What is the significance of the connections?*
4. *What is the significance of the layout?*

Architectural Patterns

An architectural pattern is a description of element and relation types together with a set of constraints on how they may be used

- These constraints define a set or family of architectures that satisfy them

Client-Server Architecture

- Client-server is a common architectural pattern
- Client and server two elements
- Coordination described by the protocol server uses
- Multiple clients can exist

- They exhibit known quality attributes

Reference Model

A reference model is a division of functionality together with data flow between the pieces

- Reference model is a standard decomposition of a known problem into parts that cooperatively solve the problem
- Show the maturity of a particular domain
    - What the various parts of a compiler?
    - How they interact with each other to perform a particular task?

Reference Architecture

A reference architecture is a reference model mapped onto software elements (that cooperatively implement the functionality defined in the reference model) and the data flows between them.

- A software architect must design a system that provides
    - Concurrency, portability,
    - modifiability, usability, security, and
    - that reflects consideration of the trade-offs among these needs

Why Software Architecture? I

- Communication among stakeholders
- Early design decisions
- Transferable abstraction of a system
- Other Point of Views
- Software architecture is concerned with:
    - Stakeholders
    - System-Level Structures
    - System qualities

Why Software Architecture? II

Software architecture involves:

- Understanding domains, problems and solutions
    - Making design decisions & trade-offs
    - Delivering working systems
- The crucial bridge between requirements and design

System-level Structures

- The traditional deliverable of the software architect
- Note that there are many structures
  - Functional, Deployment, Information, . . .
- Represented as a set of views
  - Separate concerns
  - Communicate effectively
  - Organize deliverables and activities

Stakeholders

Those who care if the system gets built

- Can be a positive or negative interest
- Includes people, groups and entities

The reason we build systems

- Systems are built for stakeholders
- Design decisions must reflect their needs

System Qualities

Non-functional characteristics (“-illities”)

- Performance, Security, Availability, . . .

Often crucial to stakeholders

- Slow functions don’t get used
- Unavailable systems stop the business

Also involves

- Functionality
    - Usability
    - Resilience
    - Performance
    - Reuse
    - Comprehensibility
    - Economic and technology constraints and trade-offs
    - Aesthetic concerns

Software Architecture views

- Software architecture is commonly organized in views, which are analogous to the different types of blueprints made in building architecture.
- Views are instances of viewpoints, where a viewpoint exists to describe the architecture from the perspective of a given set of stakeholders and their concerns.

Soft Architecture Views

- Logical view
- Process view
- Physical view
- Development view

Logical View

- This describes the architecturally significant elements of the architecture and the relationships between them.
- The logical view essentially captures the structure of the application using class diagrams or equivalents.

Process View

- This focuses on describing the concurrency and communications elements of an architecture.
- In IT applications, the main concerns are describing multi-threaded or replicated components, and the synchronous or asynchronous communication mechanisms used.

Physical View

- This depicts how the major processes and components are mapped on to the applications hardware.
- It might show, for example, how the database and web servers for an application are distributed across a number of server machines.

Development View

- This captures the internal organization of the software components, typically as they are held in a development environment or configuration management tool.
- For example, the depiction of a nested package and class hierarchy for a Java application would represent the development view of an architecture.

How to represent Software Architecture?

- Textual Representation
- Graphical Representation
- Architecture Description Languages (ADL)
- UML as an ADL???

Architectural Description Languages

- ADL is a set of notations, languages, standards and conventions for an architectural model.
- A language that is designed specifically for the representation and analysis of software architectures.
- It defines a formal syntax for expressing architectures and assigns a semantic interpretation.

ADLs

- Several different ADLs have been developed by different organizations, including:
    - AADL (Society of Automotive Engineers standard)
    - Wright (developed by Carnegie Mellon)
    - Acme (developed by Carnegie Mellon)
    - xADL (developed by UCI)
    - Darwin (developed by Imperial College London)

ADL vs Standard Design Notations

- ADLs differ from standard system design notations in the following manner:
    - ADLs tend to support components of larger granularity and are more often tailor-able to a specific domain.
    - ADLs (ideally) provide constructs to support reuse, rapid prototyping, connections as first class entities, and rationale for constraints imposed on architectural components and connections

ADL ...

- ADLs provide for a larger and more encompassing view of the software than standard design notations while at the same time providing, through various architectural views, enough detail for analysis and evaluation.

UML vs ADL

- Several languages for describing software architectures have been devised, but no consensus has yet been reached on which symbol-set and view-system should be adopted.
- The **UML** was established as a standard *"to model systems (and not just software)"* and thus applies to views about software architecture.

References & Further reading

- Chapter 1 of “Software Architecture and Practice” by Len Bass

Further Reading Assignment

- David Garlan and Mary Shaw (1994),”An Introduction to Software Architecture”, CMU Software Engineering Institute Technical Report CMU/SEI-94-TR-21, ESC-TR-94-21.