Architectural Activities & Design Process

Software Architecture
3rd Year – Semester 1
Lecture 4

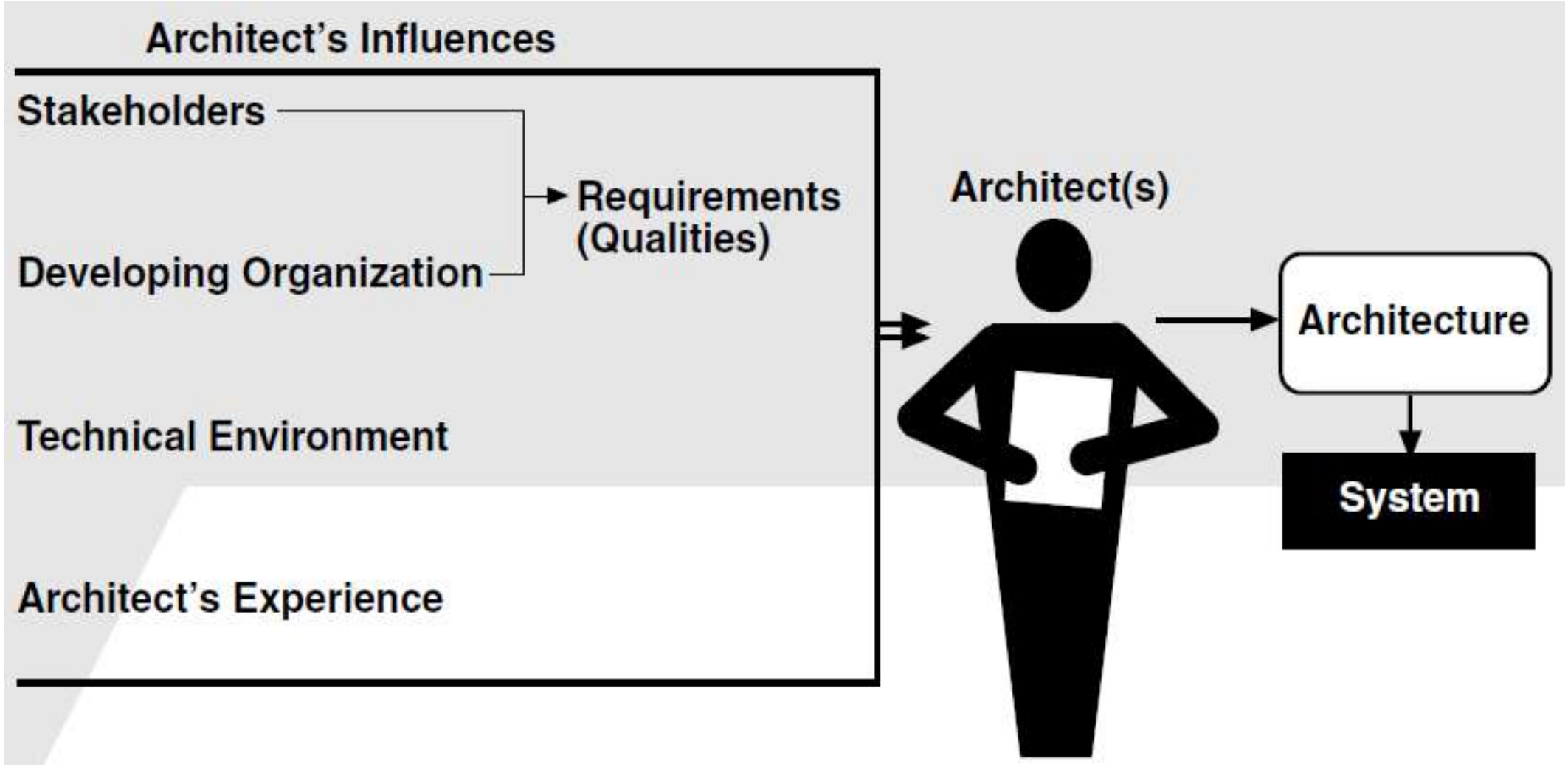
Architectural Process & the ABC

- What activities are involved in creating a Software Architecture?
- Using the selected Architecture how to realize a Design and then Implement or Manage the Evolution of a target system or application?

Where do Architectures come from?

- An architecture is the result of a set of business and technical decisions
- Influenced by:
  - Stakeholders
    - Management: Low Cost | Marketing: Time to Market | End User: UX, Security
  - Developing Organization
    - Investments already made | Future Investments | Organizational Structure
  - Experience & Background of the Architect(s)
    - Technical Skills | Domain Knowledge
  - Technical Environment
    - Software Engineering Techniques, practices & processes

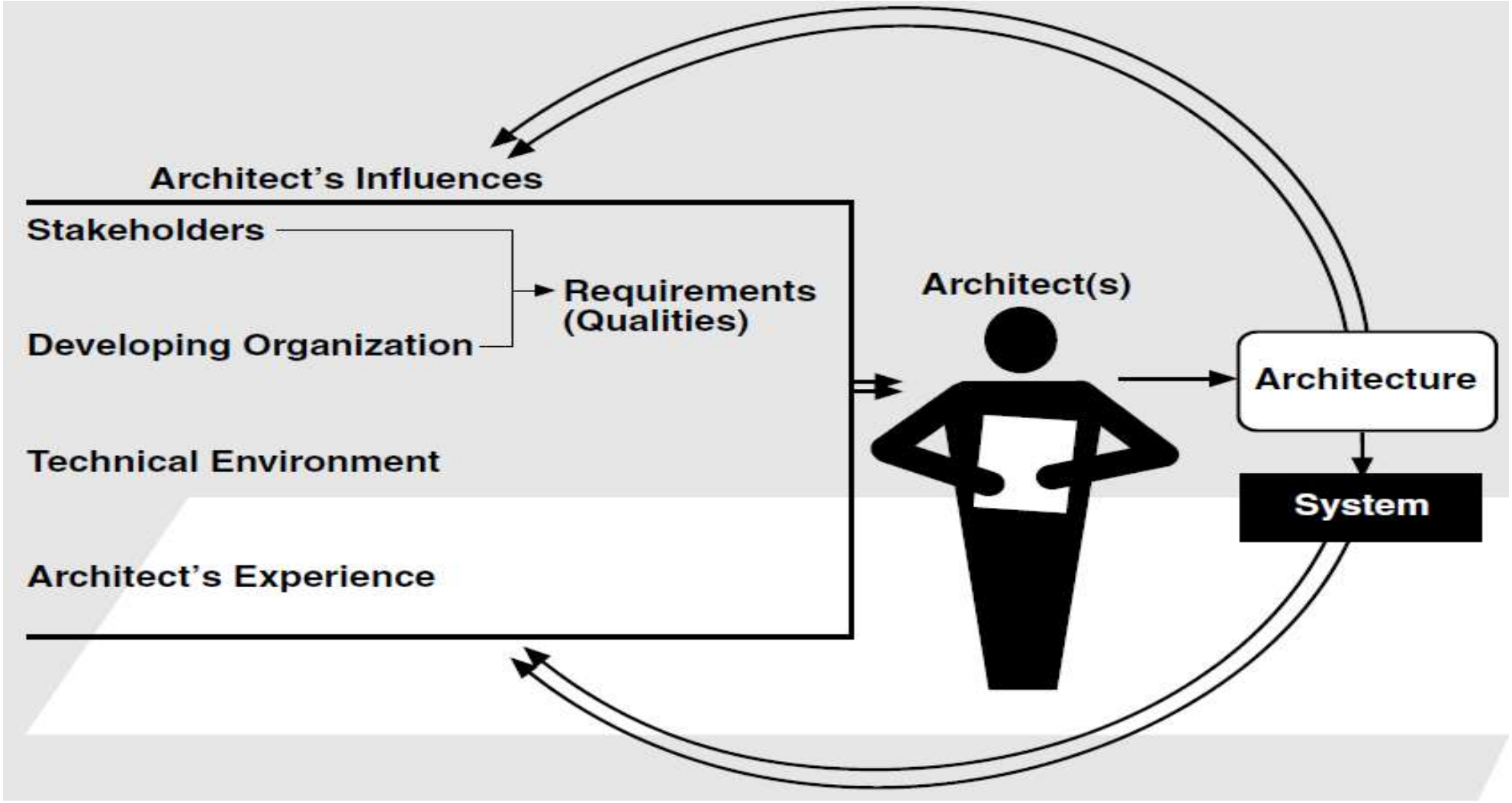
Architect's Influence



Architecture Business Cycle (ABC)

- Software architecture is a result of technical, business, and social influences.
- Its existence in turn affects the technical, business, and social environments that subsequently influence future architectures.
- We call this cycle of influences, from the environment to the architecture and back to the environment, the Architecture Business Cycle (ABC)

The Architecture Business Cycle



Ramifications of Architecture

- Affects the structure of the developing organization
  - Team Structure, Specialized Skills for Teams, etc.
- Affects the Goals of development organization
  - The organization may adjust its goals depending on the result of the current system to explore the future markets
- Affects future customer requirements
- Affects the Architect
  - Architect also will gain more experience based on the results
- Affects the Development Process & Culture

Architectural Activities

- Creating the Business Case for the System
- Understanding the Requirements
- Creating or Selecting the Architecture
- Documenting and Communicating the Architecture
- Analyzing or Evaluating the Architecture
- Implementing the system based on the Architecture
- Ensuring that the implementation Conforms to the Architecture

Creating the Business Case for the System

- Assess the market need for a system
- How much should the product cost?
- What is its targeted market?
- What is its targeted time to market?
- Will it need to interface with other systems?
- Are there system limitations that it must work within?

Note: Above cannot be decided solely by an architect, but if an architect is not consulted in the creation of the business case, it may be impossible to achieve the business goals!

Understanding the Requirements

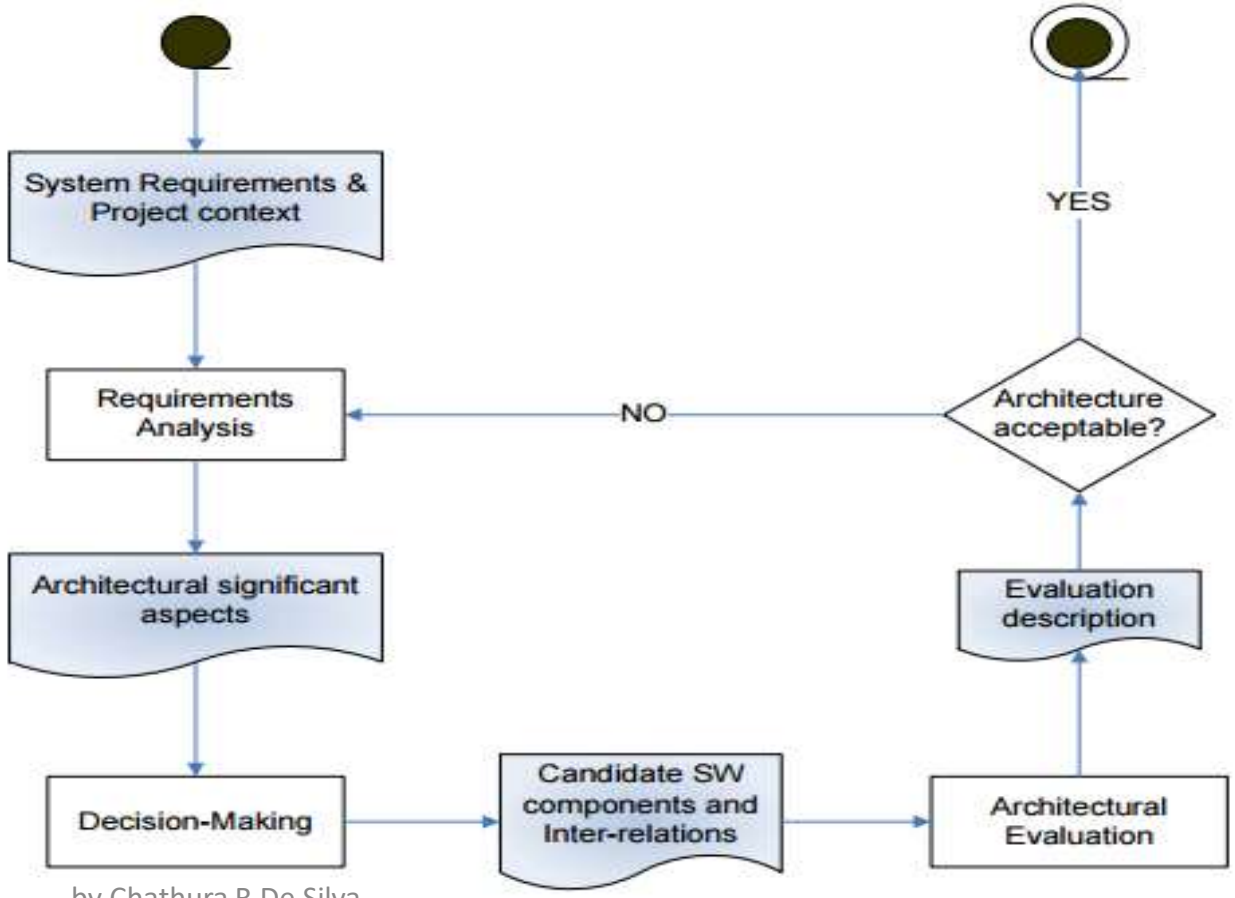
- Various methods exists for capturing functional requirements;
  - OOAD uses Scenarios or Use Cases
  - Safety Critical Systems uses Finite-state-machine models or Formal specification language
- For Non-Functional requirements;
  - Quality Attribute Scenarios
- Prototyping
- Domain Modeling

Creating or Selecting the Architecture

- The process of how to create an architecture to achieve its behavioral and quality requirements

🔧 Step-by-Step Process

Step	Description	Example
1. Review Requirements	Understand functional and non-functional requirements clearly.	The app should support 1 million users and respond in 1 second.
2. Choose Patterns	Select suitable architectural patterns (like Layered, Microservices, etc.)	Use Microservices for a large scalable system.
3. Define Components	Decide what main parts (modules, services) the system will have.	A payment service, a user service, etc.
4. Define Interactions	Specify how components communicate (e.g., REST API, message queues).	Services interact via RESTful APIs.
5. Use Existing Architectures	If a similar system was built before, reuse its architecture or parts of it.	Reuse a proven e-commerce architecture.
6. Validate Against Quality Goals	Make sure your structure meets performance, security, and other goals.	Test whether it handles peak loads.



Documenting and Communicating the Architecture

- For the architecture to be effective as the backbone of the project's design, it must be communicated clearly and unambiguously to all of the stakeholders
  - Developers: must understand the work assignments it requires of them
  - Testers: must understand the task structure it imposes on them
  - Management: must understand the scheduling implications it suggests
- Architectural documentation should be informative, unambiguous, and readable by many people with varied backgrounds

Analyzing or Evaluating the Architecture

- In any design process there will be multiple candidate designs considered, some will be rejected immediately and others will contend for primacy
- Choosing among these competing designs in a rational way is one of the architect's greatest challenges
- Methods:
  - Architecture Tradeoff Analysis Method (ATAM) is the most mature methodology
  - Cost Benefit Analysis Method (CBAM) focuses more on economic implications of architectural decisions

Implementing the system based on the Architecture

- Ensure the developers are kept faithful to the structures and interaction protocols constrained by the architecture
- The Environment should assists developers in creating and maintaining the architecture
- Architect's involvements:
  - Technology & Technical Infrastructure Selection & Setup
  - Setting up Appropriate Software Engineering Processes
  - Creating the Team
    - Identify Technical specialists
    - Identify Skill Gaps and mitigate (trainings, etc.)

Ensuring that the implementation Conforms to the Architecture

- Constant vigilance is required to ensure that the actual architecture and its representation remain faithful to each other in the maintenance phase

What makes a Good Architecture?

- No such thing as an inherently good or bad architecture
- Architectures are more or less fit for some stated purpose
- Architectures can be evaluated
  - One of the great benefits of paying attention to them
  - Should be evaluated only in the context of specific goals
- Rules of Thumb (from knowledge & experience):
  - Process Recommendations:
    - Functional Requirements & identifying Quality Attributes which are of high priority
    - Analyze & formally evaluate before it is too late to change
  - Product Recommendations:
    - Software Principles e.g. Information hiding, Separation of concerns
    - Write tasks or processes to allow easy reallocation, perhaps at runtime

Architectural Design Process

- Objectives of the Design Process:
  - Creativity
    - Enhance your skillset
    - Provide new tools
  - Method
    - Focus on highly effective techniques
  - Develop judgment: when to develop novel solutions, and when to follow established method

Engineering Design Process

- Feasibility:
  - Identifying a set of feasible concepts for the design as a whole
- Preliminary design:
  - Selection and development of the best concept
- Detailed design:
  - Development of engineering descriptions of the concept
- Planning:
  - Evaluating and altering the concept to suit the requirements of production, distribution, consumption and product retirement

Potential Problems

- If the designer is unable to produce a set of feasible concepts, progress stops
- As problems and products increase in size and complexity, the probability that any one individual can successfully perform the first steps decreases
- The standard approach does not directly address the situation where system design is at stake, i.e. when relationship between a set of products is at issue
- As complexity increases or the experience of the designer is not sufficient, alternative approaches to the design process must be adopted

Alternative Design Strategies

- Standard
  - Linear model
- Cyclic
  - Process can revert to an earlier stage
- Parallel
  - Independent alternatives are explored in parallel
- Adaptive (“lay tracks as you go”)
  - The next design strategy of the design activity is decided at the end of a given stage
- Incremental
  - Each stage of development is treated as a task of incrementally improving the existing design

Identifying a Viable Strategy

- Use fundamental design tools: abstraction and modularity
  - But how? Top-down: Start with broad system goals (e.g., "real-time updates") and decompose into subsystems.
    Bottom-up: Reuse proven components (e.g., authentication modules) to build upward
- Inspiration, where inspiration is needed. Predictable techniques elsewhere.
  - But where is creativity required?
- Applying own experience or experience of others

Tools & Patterns of Software Engineering

- Abstraction
  - Abstraction(1): look at details, and abstract “up” to concepts
  - Abstraction(2): choose concepts, then add detailed substructure, and move “down”
- Separation of Concerns
- Architectural Patterns & Styles
  - Layered
  - Model View Controller (MVC)
  - Client-Server

Controlling the Design Strategy

- Manage the Activity: Exploring diverse approaches to the problem demands that some care be used in managing the activity
- Review: Identify and review critical decisions
- Cost: Relate the costs of research and design to the penalty for taking wrong decisions
- Enforce: Insulate uncertain decisions
- Cross Check with Requirements: Continually re-evaluate system requirements in light of what the design exploration yields

Insights from Requirements

- In many cases new architectures can be created based upon experience with and improvement to pre-existing architectures
- Requirements can use a vocabulary of known architectural choices and therefore reflect experience
- The interaction between past design and new requirements means that many critical decisions for a new design can be identified or made as a requirement

Insights from Implementation

- Constraints on the implementation activity may help shape the design
- Externally motivated constraints might dictate
  - Use of a middleware
  - Use of a particular programming language
  - Software reuse
- Design and implementation may proceed cooperatively
- Initial partial implementation activities may yield critical performance or feasibility information

References

- Software Architecture in Practice, 2nd Ed., by Len Bass, Paul Clements, Rick Kazman
- http://www.ece.ubc.ca/~matei/EECE417/BASS/ch01.html
- http://www.ics.uci.edu/~taylor/classes/211/DesignAndArchitecture.pdf