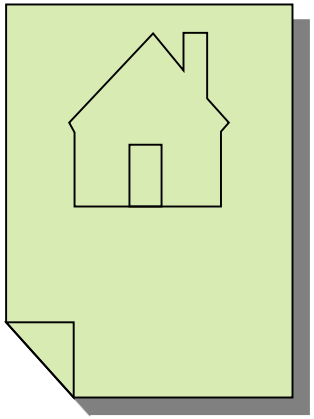
SE 4352
Software Architecture and Design

Fall 2018

Module 1

What is software architecture?

Analogy From Building Architecture



What can we learn from buildings?

- A building HAS an architecture
  - Separate but linked to the physical structure
  - We can talk about it and describe it and compare it
    - Major elements
    - Composition
    - Arrangement
- Architecture is induced by the design to meet needs
- Distinctive role and character of an architect
- Process is not as important as architecture
- Architecture has matured over the years into discipline

Limitations of the analogy

- Almost everyone knows a lot about buildings
- You can discern a building's architecture by looking at it
- Software isn't restricted by physical limitations like buildings
- The software construction industry is not developed to the same degree as the building construction industry
- Software is dynamic

Software Architecture

- In which era were the first buildings made?

  Mesopotamia, Greece

  250 BC

Architecture is Early

- Architecture represents the set of earliest design decisions
  - Hardest to change
  - Most critical to get right
- Architecture is the first design artifact where a system's quality attributes are addressed

Software Architecture defined

Software architecture is what software architects do!!!!!

Software Architecture: Formal Definitions

- ANSI/IEEE 1471-2000
  - Software architecture is the **fundamental organization** of a system, embodied in its **components**, their **relationships** to each other and the environment, and the **principles** governing its design and evolution
- ACM
  - "…architecture is concerned with the selection of architectural **elements**, their **interactions**, and the constraints on those elements and the interactions necessary to provide a framework in which to satisfy the requirements and serve as a basis for the design"
- SEI
  - *"The 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."*

Some definitions from experts

Booch, Kruchten, Reitman, Bittner, and Shaw

> Software architecture encompasses the set of significant decisions about the organization of a software system
>
> - Selection of the structural elements and their interfaces by which a system is composed
> - Behavior as specified in collaborations among those elements
> - Composition of these structural and behavioral elements into larger subsystems
> - Architectural style that guides this organization

Boehm

> A software system architecture comprises
>
> - A collection of software and system components, connections, and constraints
> - A collection of system stakeholders' need statements
> - A rationale which demonstrates that the components, connections, and constraints define a system that, if implemented, would satisfy the collection of system stakeholders' need statements

Clements

> 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

Garlan and Shaw

> [Software architecture goes] beyond the algorithms and data structures of the computation; designing and specifying the overall system structure emerges as a new kind of problem. Structural issues include gross organization and global control structure; protocols for communication, synchronization, and data access; assignment of functionality to design elements; physical distribution; composition of design elements; scaling and performance; and selection among design alternatives.

Common elements of these definitions

- Architecture defines major components
- Architecture defines component relationships (structures) and interactions
- Architecture omits content information about components that does not pertain to their interactions
- Behavior of components is a part of architecture in so far as it can be discerned from the point of view of another component
- Every system has an architecture (even a system composed of one component)
- Architecture defines the rationale behind the components and the structure
- Architecture is not a single structure -- no single structure is the architecture

Software Architecture

- *Software architecture* is the process of designing the global organization of a software system, including:
  - Dividing software into subsystems.
  - Deciding how these will interact.
  - Determining their interfaces.
    - The architecture is the core of the design, so all software engineers need to understand it.
    - The architecture will often constrain the overall efficiency, reusability and maintainability of the system.

Architecting a dog house

Architecting a house

Architecting a high rise

Architecture characteristics

- Performance
  - Localize critical operations and minimize communications. Use large rather than fine-grain components.
- Security
  - Use a layered architecture with critical assets in the inner layers.
- Safety
  - Localize safety-critical features in a small number of sub-systems.
- Availability
  - Include redundant components and mechanisms for fault tolerance.
- Maintainability
  - Use fine-grain, replaceable components.

Design *stable* architecture

- To ensure the maintainability and reliability of a system, an architectural model must be designed to be *stable*.
  - Being stable means that the new features can be easily added with only small changes to the architecture

Contents of a good architectural model

- A system's architecture will often be expressed in terms of several different *views*
  - The logical breakdown into subsystems
  - The interfaces among the subsystems
  - The dynamics of the interaction among components at run time
  - The data that will be shared among the subsystems
  - The components that will exist at run time, and the machines or devices on which they will be located

Developing an architectural model

- Start by sketching an outline of the architecture
  - Based on the principal requirements and use cases
  - Determine the main components that will be needed
  - Choose among the various architectural styles
  - *Suggestion*: have several different teams independently develop a first draft of the architecture and merge together the best ideas

Developing an architectural model

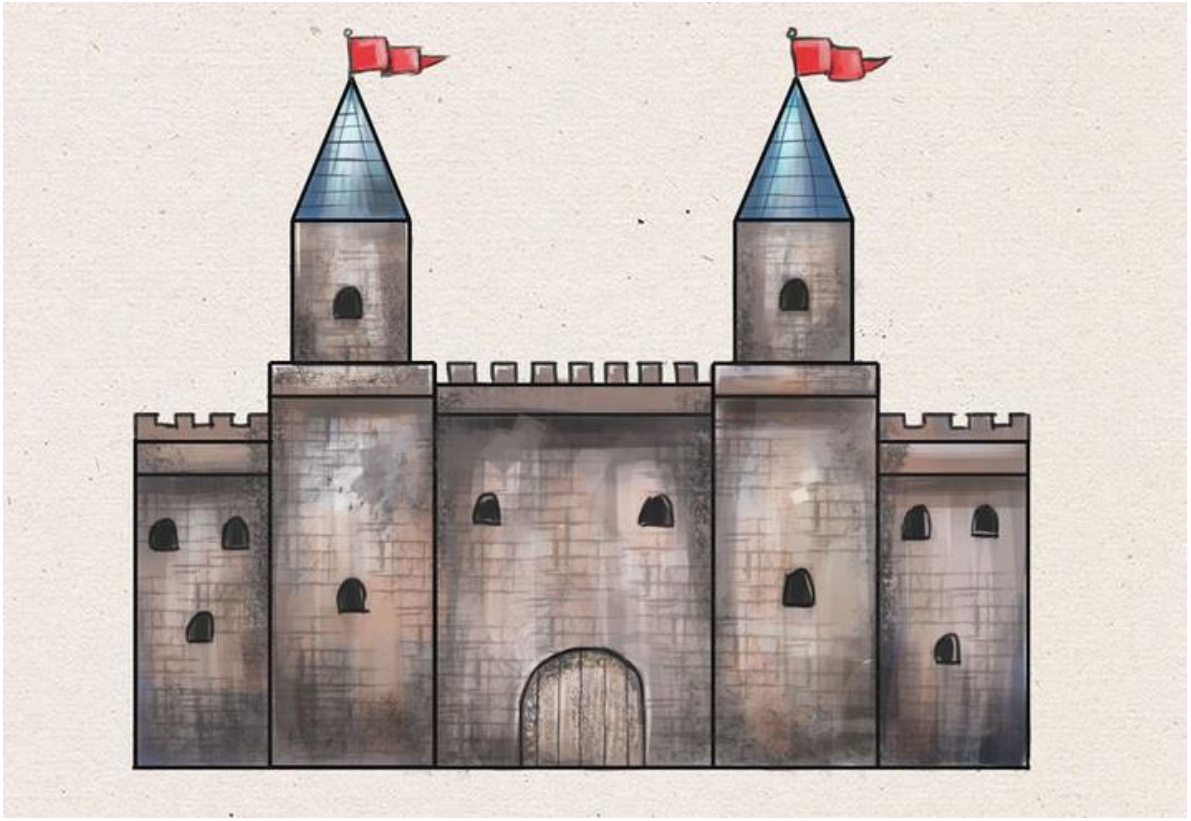
- Refine the architecture
  - Identify the main ways in which the components will interact and the interfaces between them
  - Decide how each piece of data and functionality will be distributed among the various components
  - Determine if you can re-use an existing framework, if you can build a framework
- Consider each use case and adjust the architecture to make it realizable
- Mature the architecture

Architectural Styles

- The notion of patterns can be applied to software architecture.
  - These are called *architectural patterns* or *architectural styles*.
  - Each allows you to design flexible systems using components
    - The components are as independent of each other as possible.

Architectural Styles

- Set of constraints placed on development in order to produce desirable qualities Eg. Medieval castle



Software Development Models

- Software development life cycle (SDLC)
- Structure imposed on the development of a software product
- Several models for such processes
- Waterfall, Iterative, Spiral
- Each describes approaches to a variety of tasks or activities that take place during the process

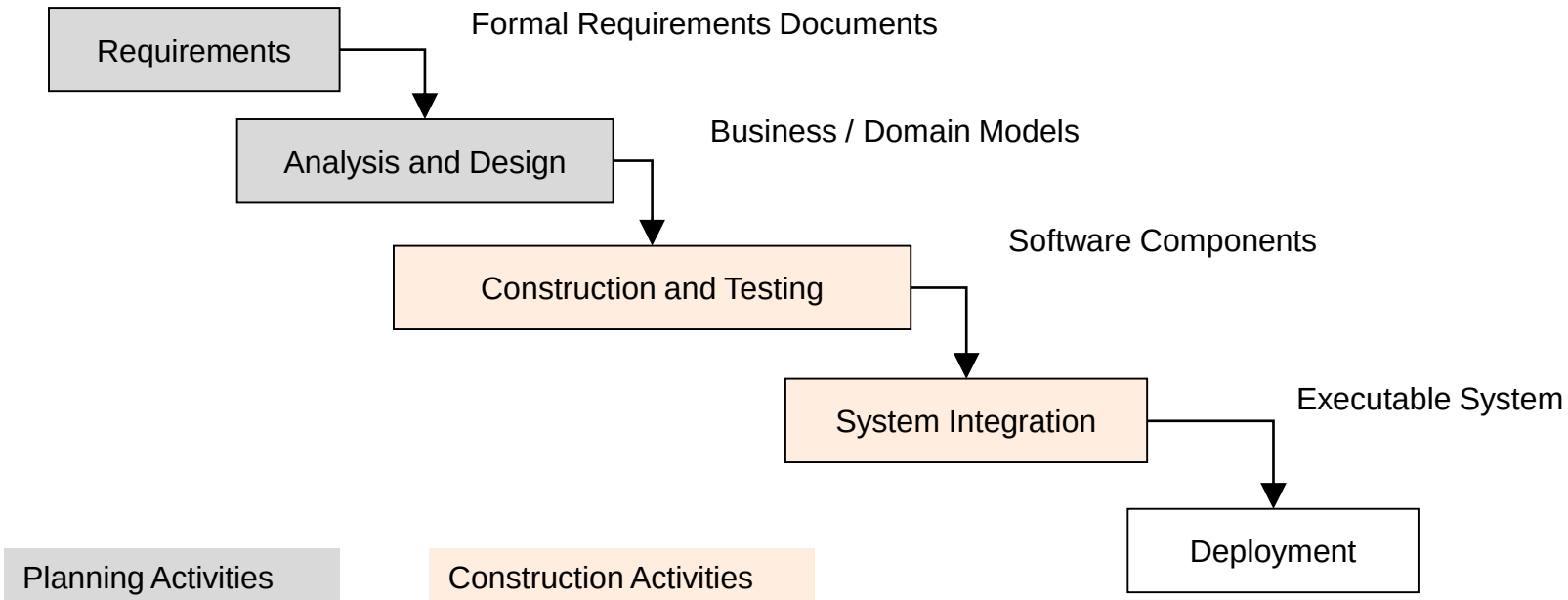
Traditional Waterfall Software Development Model

- Formal Description - 1970 by Winston W. Royce
- Owes its origin to the standard workflow process in construction and manufacturing industries

Software Development Process:

The Waterfall Process

- Each rectangle represents an activity in the overall process.
- Each activity produces work products that are used by activities in later stages of the process.



Iterative Software Development Process

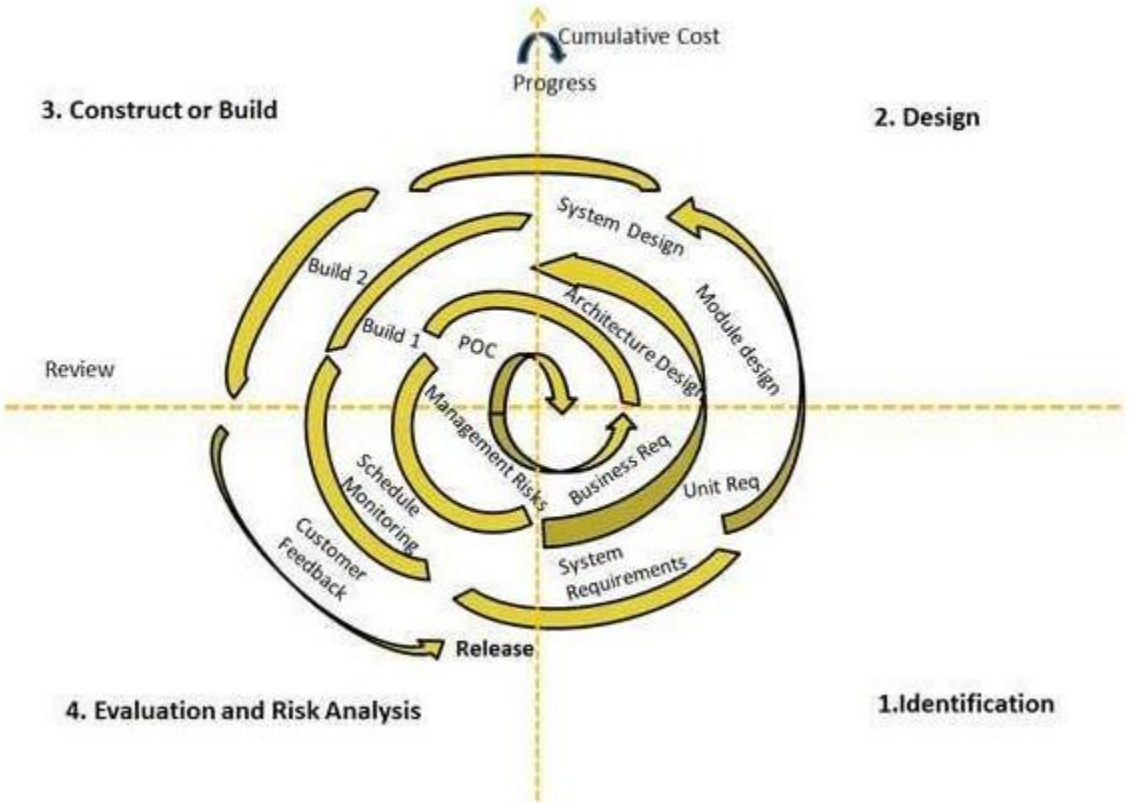
- An iterative software development process is a process that minimizes the damage caused by new, changing, missing, or incorrectly understood system requirements.
- An iterative process accomplishes this by:
  - Recognizing that the system can not be completely understood before construction begins.
  - Defining, designing, and constructing the system in many incremental deliveries.
  - Allowing the customer to evaluate each incremental delivery and to make necessary changes.

Agile Software Development Model

- Adaptability and Agility, requirements changing constantly
- People-oriented rather than process-oriented
- Extreme Programming (XP), Scrum - examples of Agile methodology
- Iterations vary from three weeks to a month

Spiral Development Model

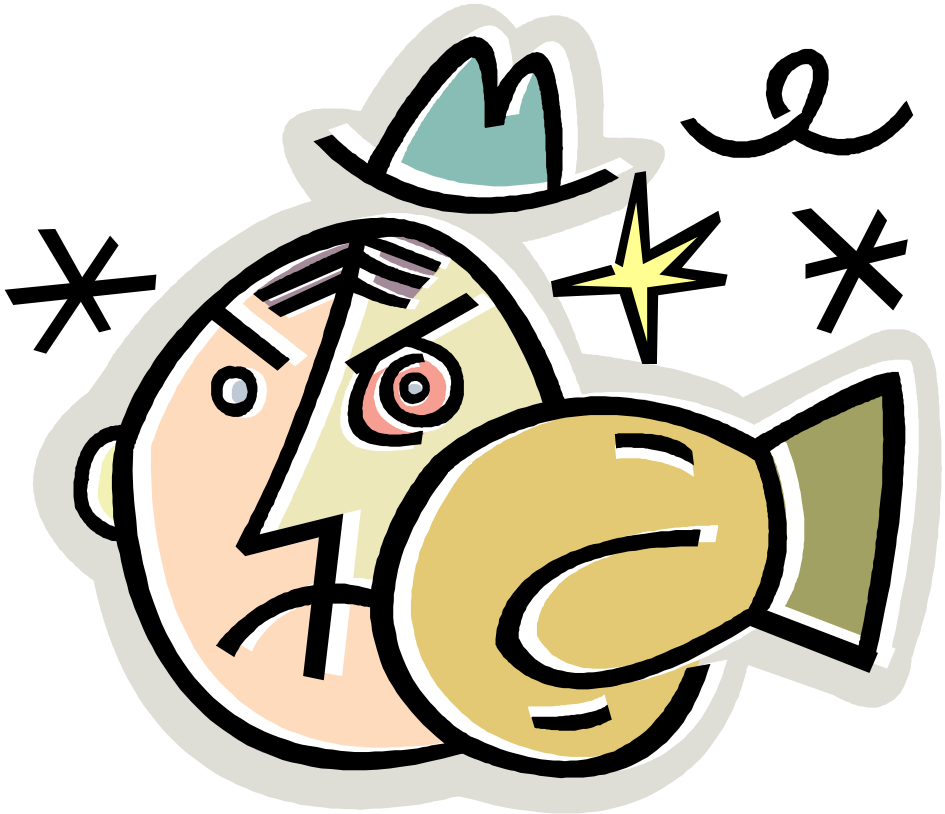
- Combination of iterative development process model and waterfall model, very high emphasis on risk analysis
- Incremental releases of the product, or incremental refinement through each iteration around the spiral
- Four phases: Identification, Design, Construct or Build, Evaluation and Risk Analysis



Software Development Model

- What are the pros and cons of the traditional waterfall, agile, and spiral software development models?

Architecture vs design



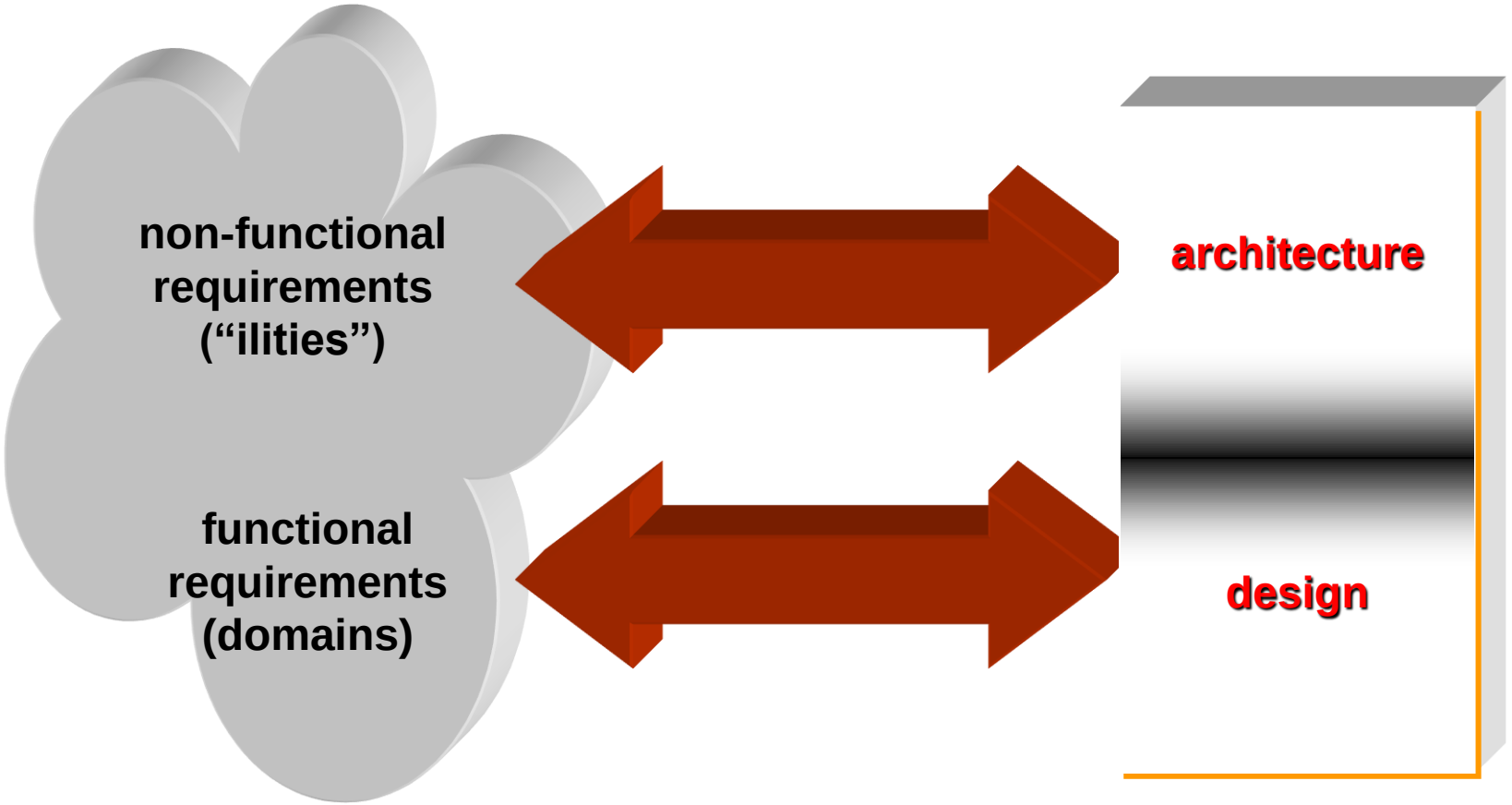
What is Software Architecture vs design?

- It’s about software design
  - All architecture is software design, but not all design is software architecture
  - Part of the design process
- Simply, architecture focuses on ‘issues that will be difficult/impossible to change once the system is built’
  - Quality attributes like security, performance
  - Non-functional requirements like cost, deployment hardware

Architecture vs. Design

Architecture: where non-functional decisions are cast, and functional requirements are partitioned

Design: where functional requirements are accomplished



Important : this is a general guideline – sometimes the borders are blurred

SE Design vs Software Architecture

- Scope
- Software Architecture
  - Broad and shallow
  - Describes overall system
- Software Engineering Design
  - Narrow and deep
  - Focused on single components

More

- Software Architecture
  - High level design
  - Main use cases of the system
  - Programming language independent
  - Overall structure
  - Hard to change
- Detailed Design
  - Low level design
  - Inner structure of the main modules
  - May consider what language is used
  - Detailed enough to be programmed