投票 最多可选1项

设置

Are you ready?

A Yes

B No

提交

Software Engineering

Part 2
Modeling

Chapter 13
Architectural Design

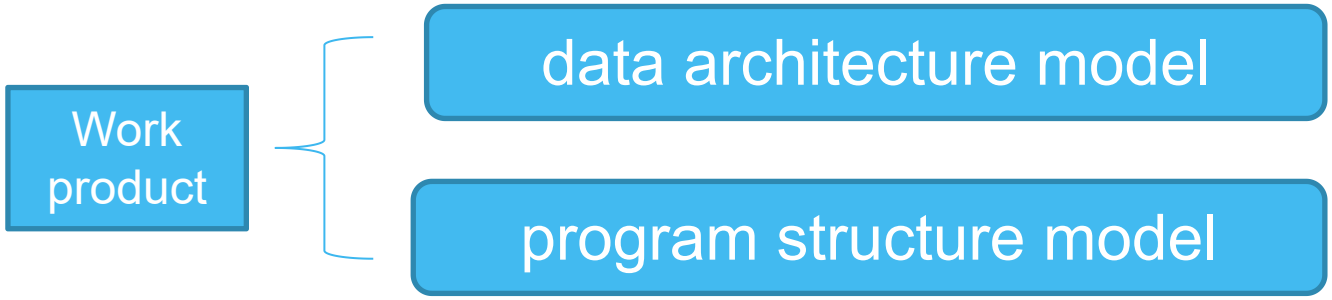
Contents

Contents

13.1 What is Architecture?

Architecture Defination [Bas03]:

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.



13.1 Why Architecture?

Six ways to use the architectural models:

- to understand the system
- to determine amount of reuse from other systems and the reusability of the system being designed
- to provide blueprint for system construction
- to reason about system evolution
- to analyze dependencies
- to support management decisions and understand risks

13.1 Architectural Decision Description

INFO

Architecture Decision Description Template

Each major architectural decision can be documented for later review by stakeholders who want to understand the architecture description that has been proposed. The template presented in this sidebar is an adapted and abbreviated version of a template proposed by Tyree and Ackerman [Tyr05].

Design issue: Describe the architectural design issues that are to be addressed.

Resolution: State the approach you've chosen to address the design issue.

Category: Specify the design category that the issue and resolution address (e.g., data design, content structure, component structure, integration, presentation).

Assumptions: Indicate any assumptions that helped shape the decision.

Constraints: Specify any environmental constraints that helped shape the decision (e.g., technology standards, available patterns, project-related issues).

Alternatives: Briefly describe the architectural design alternatives that were considered and why they were rejected.

Argument: State why you chose the resolution over other alternatives.

Implications: Indicate the design consequences of making the decision. How will the resolution affect other architectural design issues? Will the resolution constrain the design in any way?

Related decisions: What other documented decisions are related to this decision?

Related concerns: What other requirements are related to this decision?

Work products: Indicate where this decision will be reflected in the architecture description.

Notes: Reference any team notes or other documentation that was used to make the decision.

主观题 10分

设置

What is the difference/relationship between architecture and design?

architecture: class

design: instance

正常使用主观题需2.0以上版本雨课堂

作答

主观题 10分

设置

What is the difference/relationship between architecture and design?

architecture: class

design: instance

正常使用主观题需2.0以上版本雨课堂

作答

13.1 Architectural Genres

- *Genre* : Within each category, you encounter a number of subcategories.
  - For example, within the genre of *buildings*, you would encounter the following general styles: houses, apartment buildings, office buildings, industrial building, warehouses, and so on.
  - Within each general style, it would have a structure that can be described using a set of predictable patterns.

apartment buildings

office buildings

industrial building

13.2 Architectural Styles

Each style describes a system category that encompasses:

(1) a **set of components** (e.g., a database, computational modules) that perform a function required by a system,

(2) a **set of connectors** that enable "communication, coordination and cooperation" among components,

(3) **constraints** that define how components can be integrated to form the system

(4) **semantic models** that enable a designer to understand the overall properties of a system by analyzing the known properties of its constituent parts.

13.2 Architectural Styles

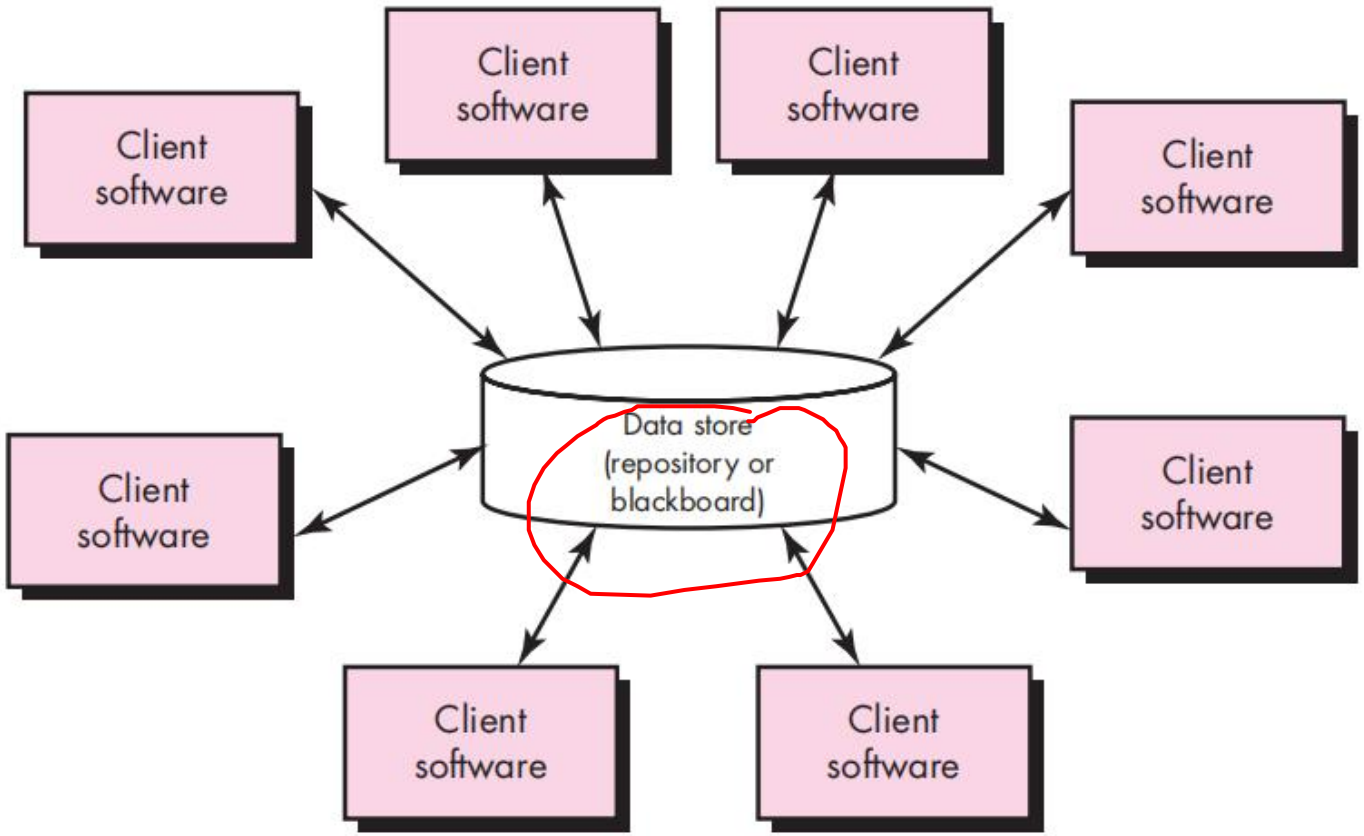
David Garlan

Mary Shaw

An Introduction to Sofware Architecture. Carnegie Mellon University,1994

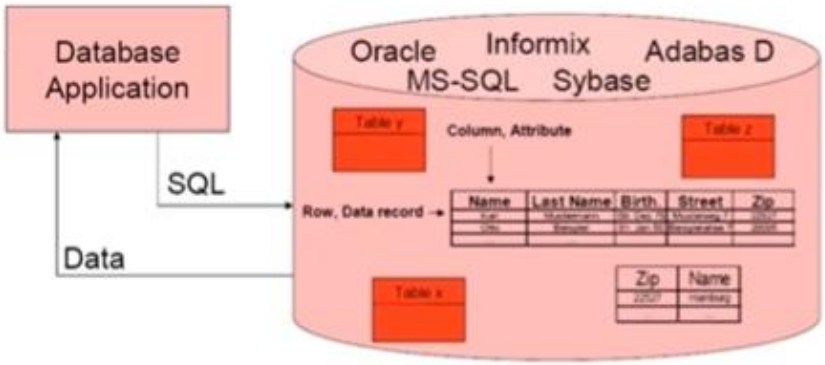
- Data-centered architectures
- Data flow architectures
- Call and return architectures
- Object-oriented architectures
- Layered architectures

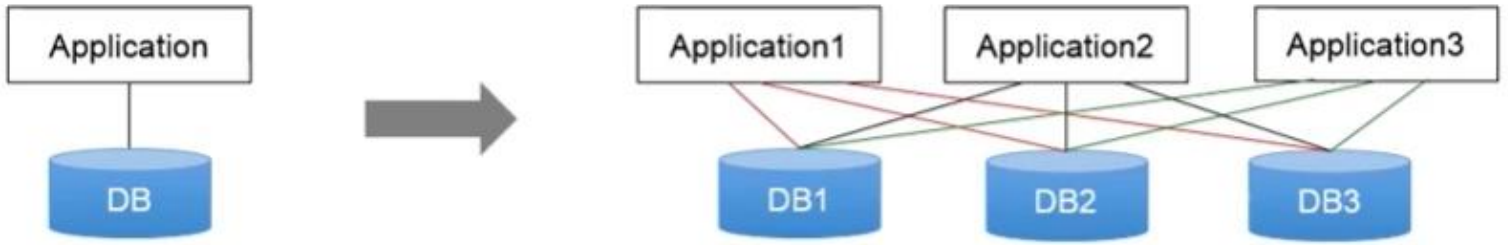
13.3.1 Data-Centered Architecture



13.3.1 Data-Centered Architecture

- Example: Database application system

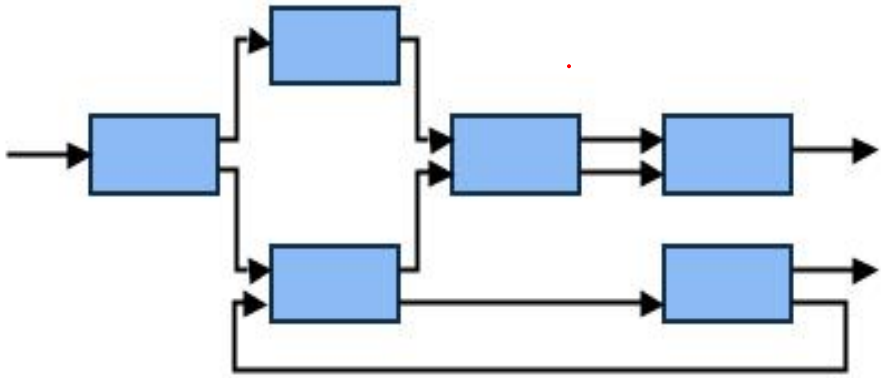


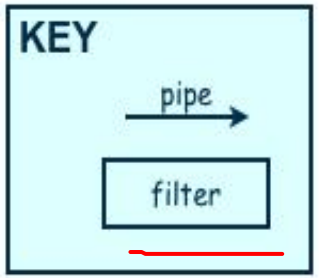


Cite from: software enginnering, https://www.bilibili.com/video/av88249030?p=66

13.3.1 Data Flow Architecture

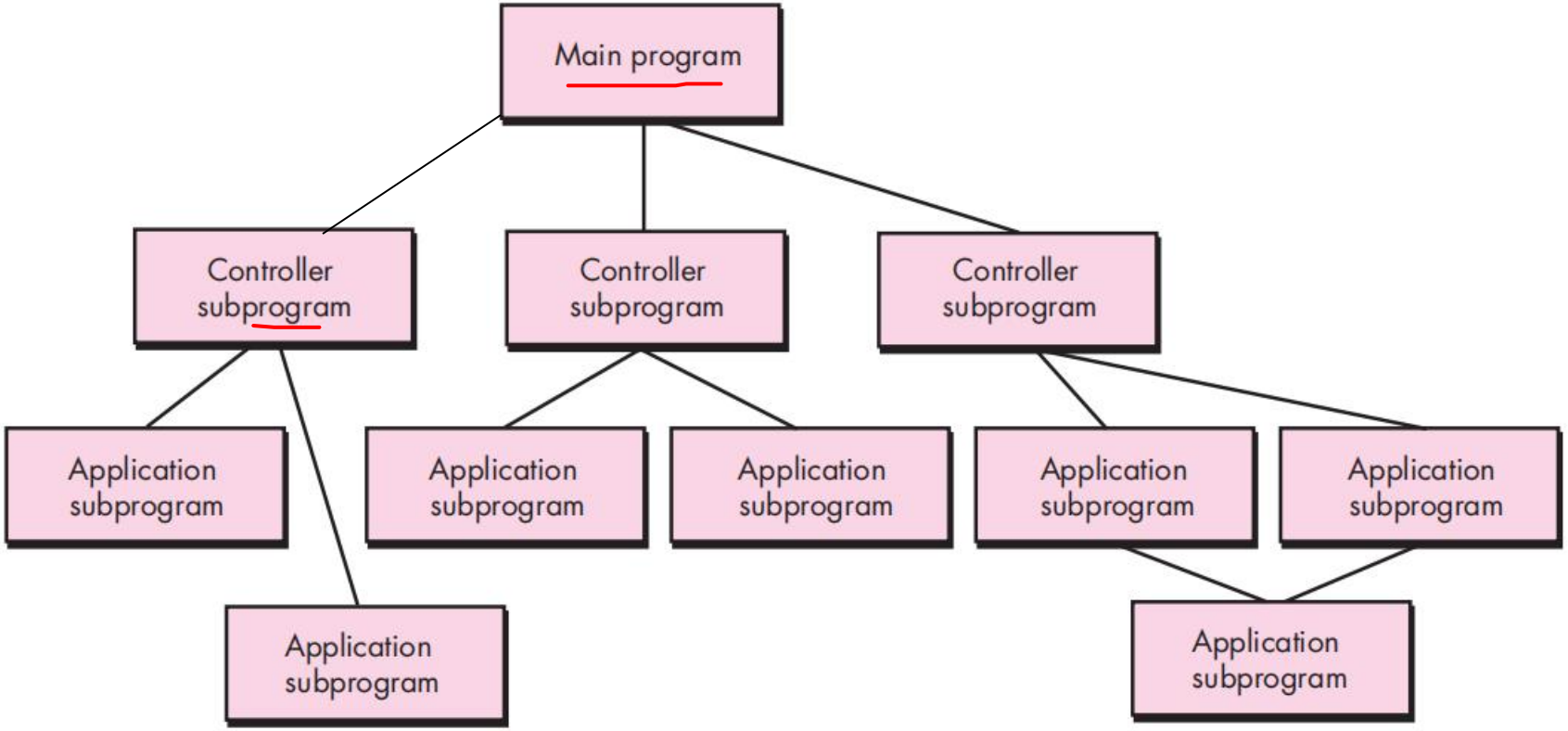
- The system has
  - Streams of data (pipe) for input and output
  - Transformation of the data (filter)





- Three-pass compiler

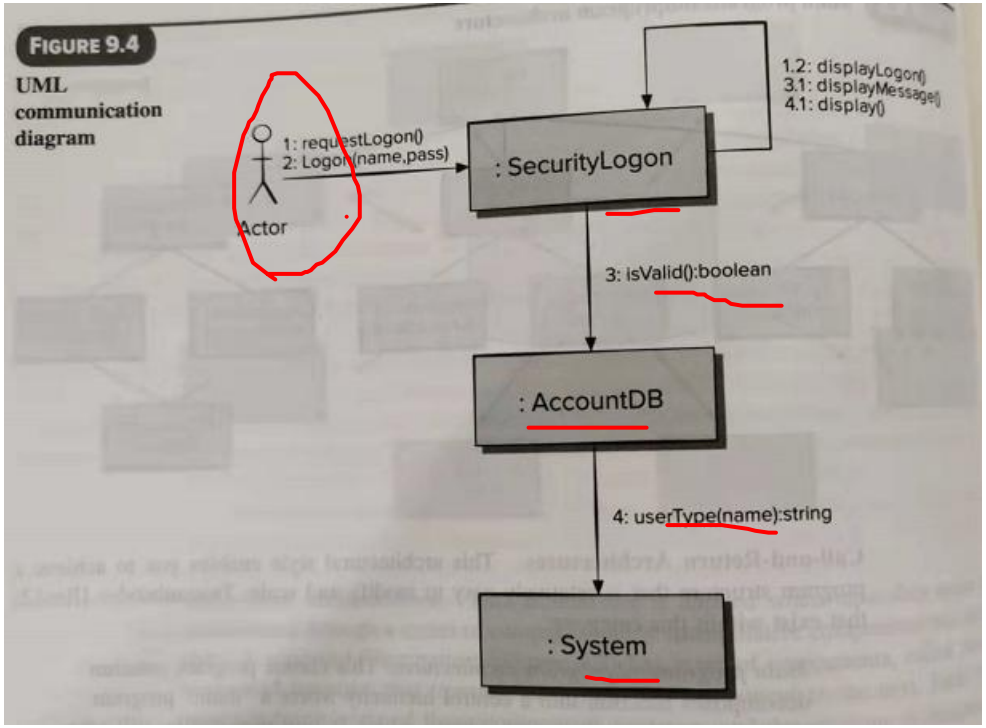
13.3.1 Call and Return Architecture



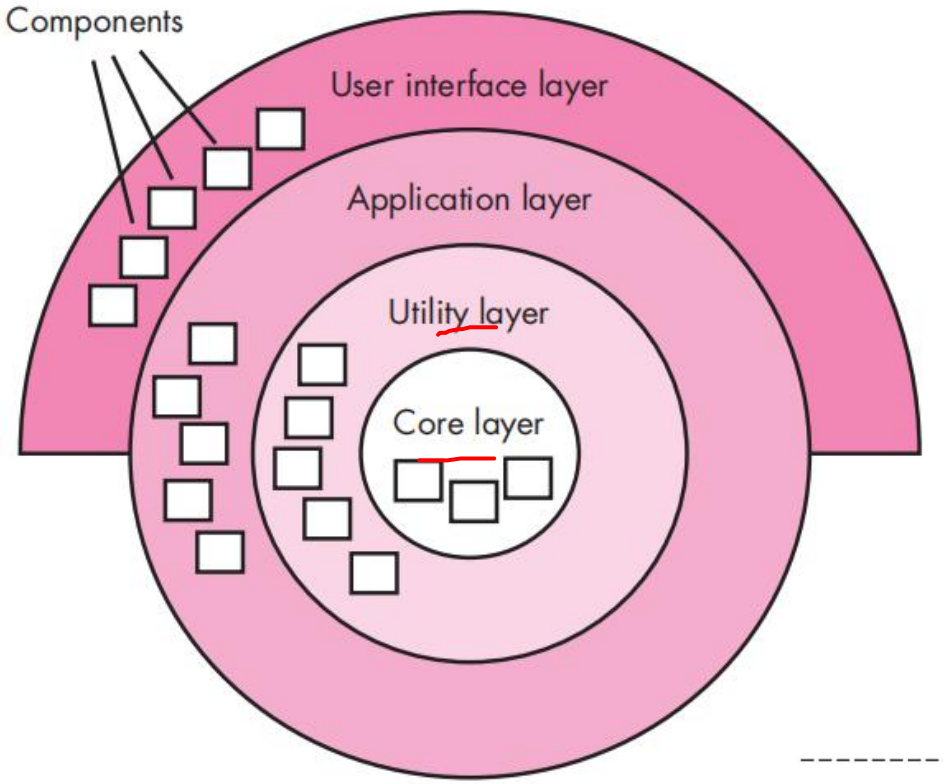
- Example: C program

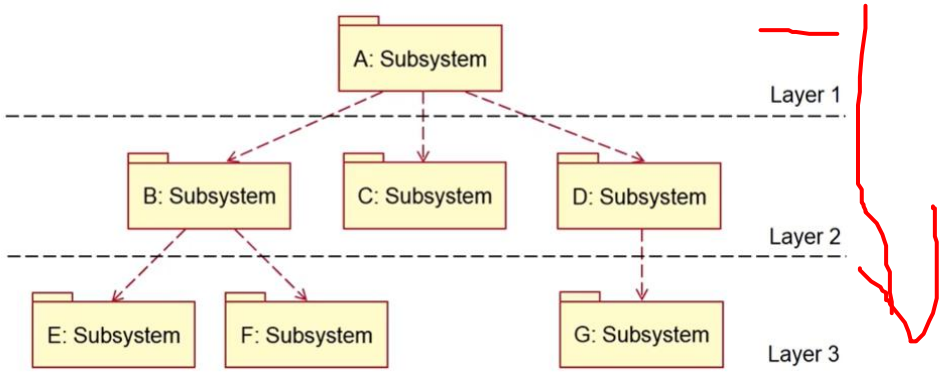
13.3.1 Object-Oriented Architectures

- Components:
  - class: encapsulate data and the operations
  - communication and coordination between components: message



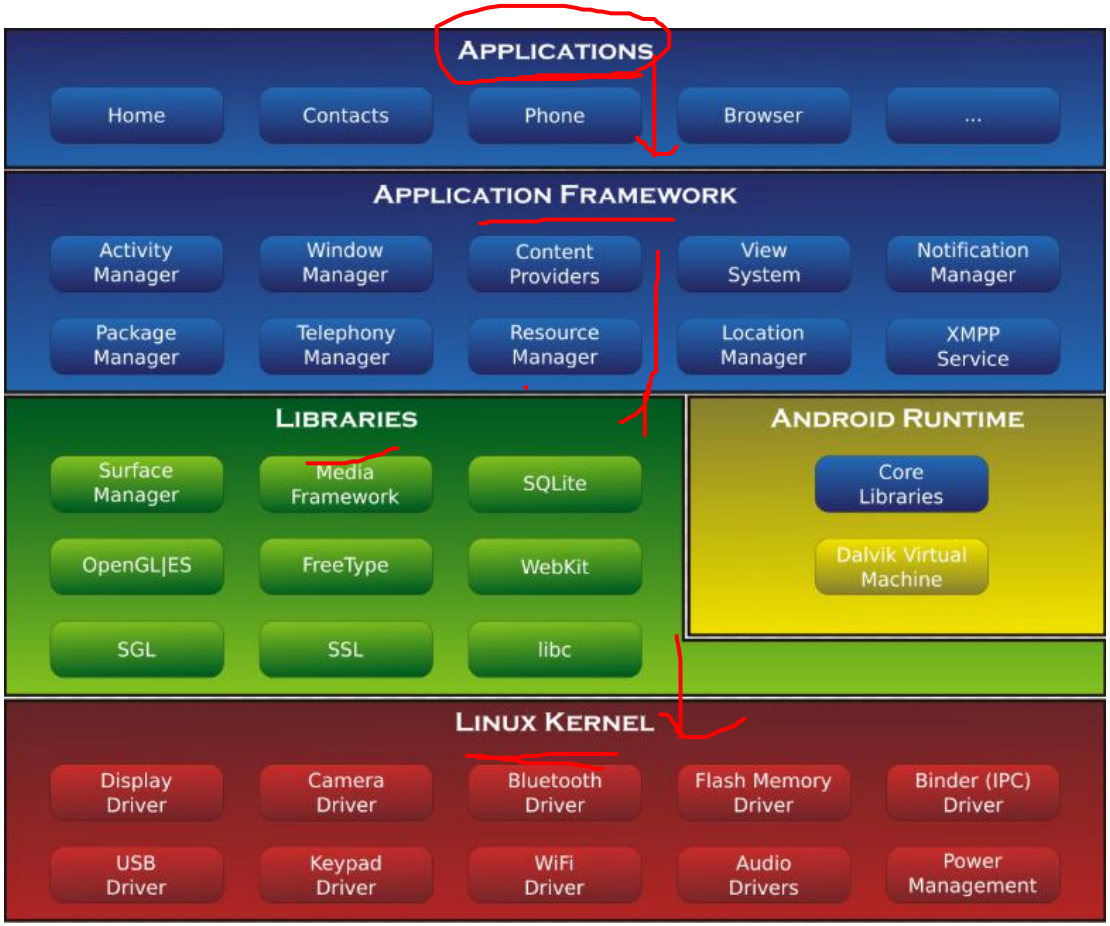
13.3.1 Layered Architecture





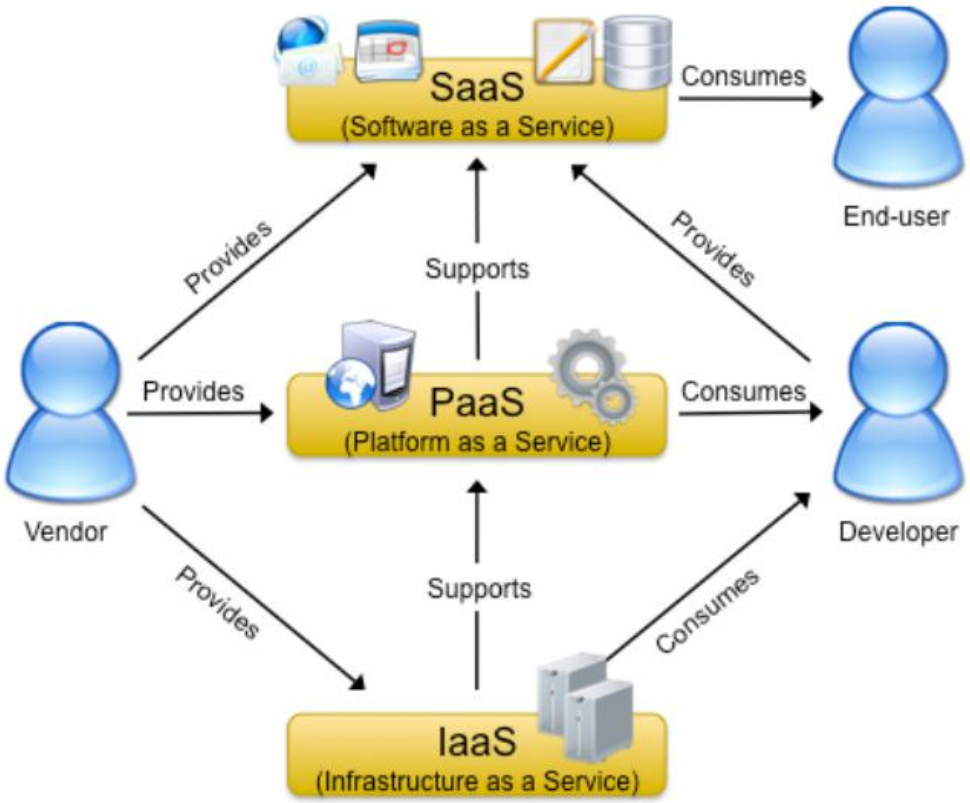
13.3.1 Layered Architecture

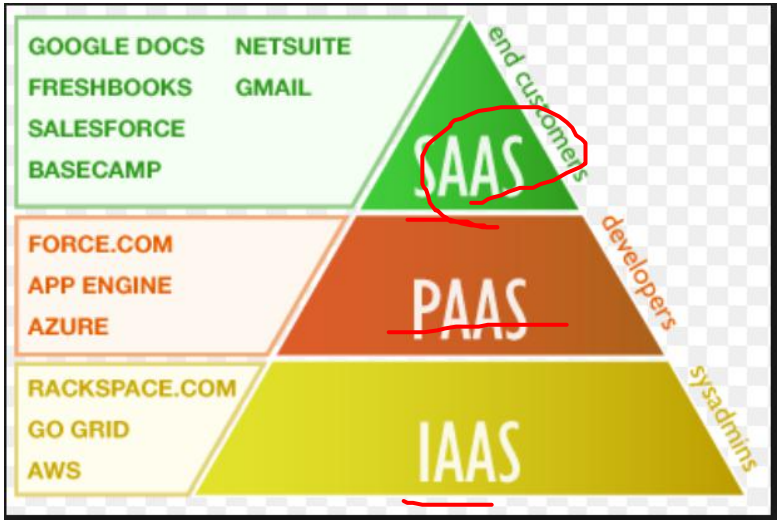
- android operating system



13.3.1 Layered Architecture

- The Cloud Computing System

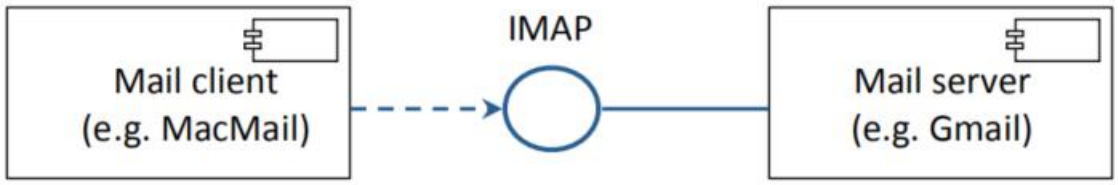




13.3.1 Client-Server Architecture

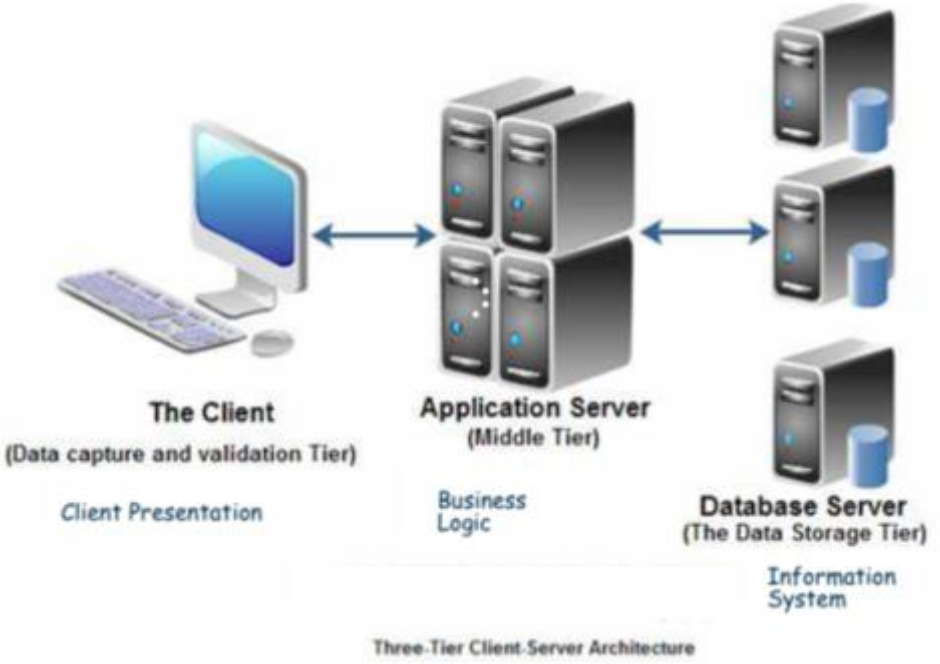
- Two types of components:
  - Server components offer services
  - Clients access them using a request/reply protocol
- Client may send the server an executable function, called a callback
  - The server subsequently calls under specific circumstances

- Mail system

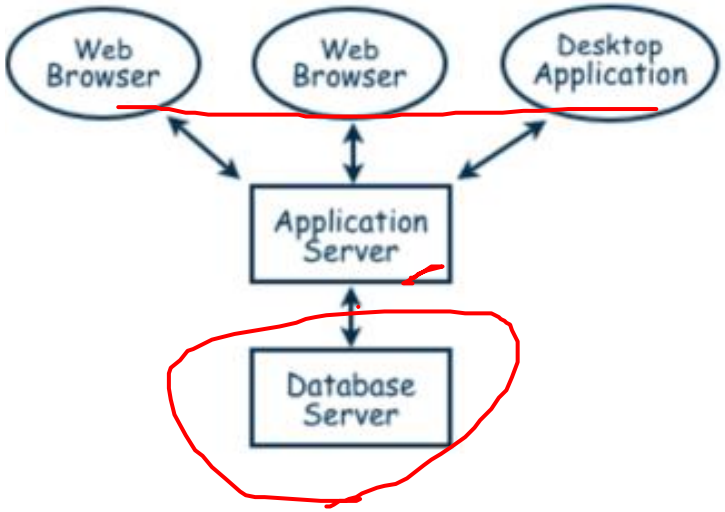


13.3.1 Client-Server Architecture

- Three-tier client-server architecture

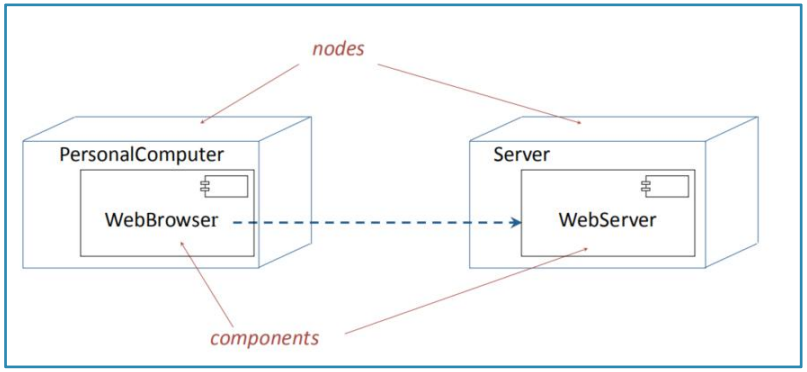


Three-Tier Client-Server Architecture

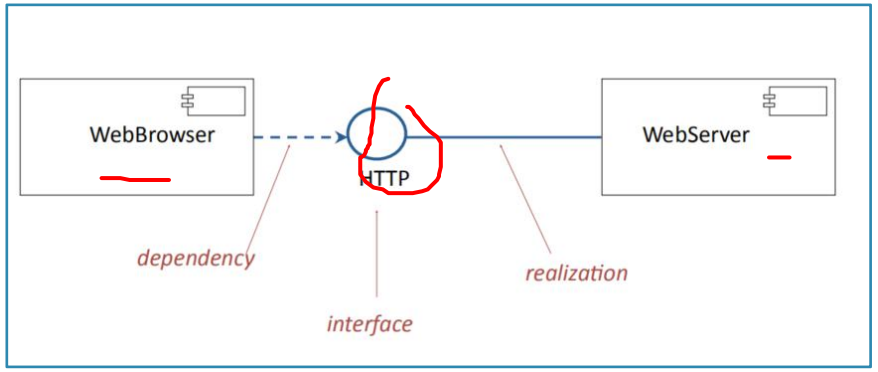


13.3.1 Architecture Design UML diagram

- UML diagrams in Architecture Design
  - Component diagram
  - Deployment diagram
  - Package diagram
  - Class diagram



Deployment diagram



Component diagram

13.3.1 Architecture Style

- How to determine your style?
  - more than one pattern might be appropriate
  - combination different styles

  (e.g. a layered style + data-centered style for many database applications)

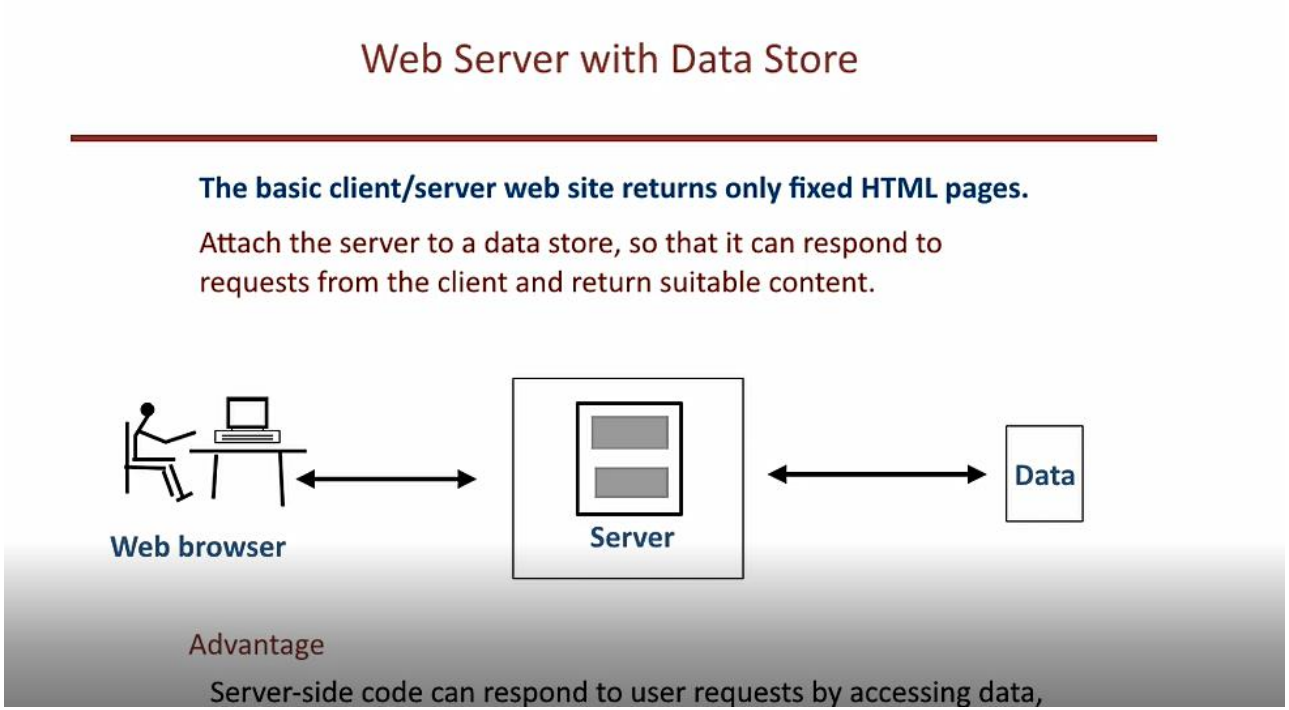
- Are there any other styles?
  - Distributed?
  - Time critical (real time) system?
  - ....

Take a break

Five minutes

13.7 Architectural Design

Architectural Design Examples: Watch a video



Cited from: CS5150 Cornel University slide

cs169: https://www.bilibili.com/video/BV1t4411W7Xq?p=5

CS164 Harvard https://www.bilibili.com/video/BV1DF411H7W7?p=3

13.3.2 Architectural 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
- 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.

13.4 Architectural Considerations

- Economy – The best software is uncluttered and relies on abstraction to reduce unnecessary detail.
- Visibility – Architectural decisions and the reasons for them should be obvious to software engineers who examine the model at a later time.
- Spacing – Separation of concerns in a design without introducing hidden dependencies.
- Symmetry – Architectural symmetry (open/close) implies that a system is consistent and balanced in its attributes.
- Emergence – Emergent, self-organized behavior and control (real system: sequence and duration of event).

13.6 Architectural Design

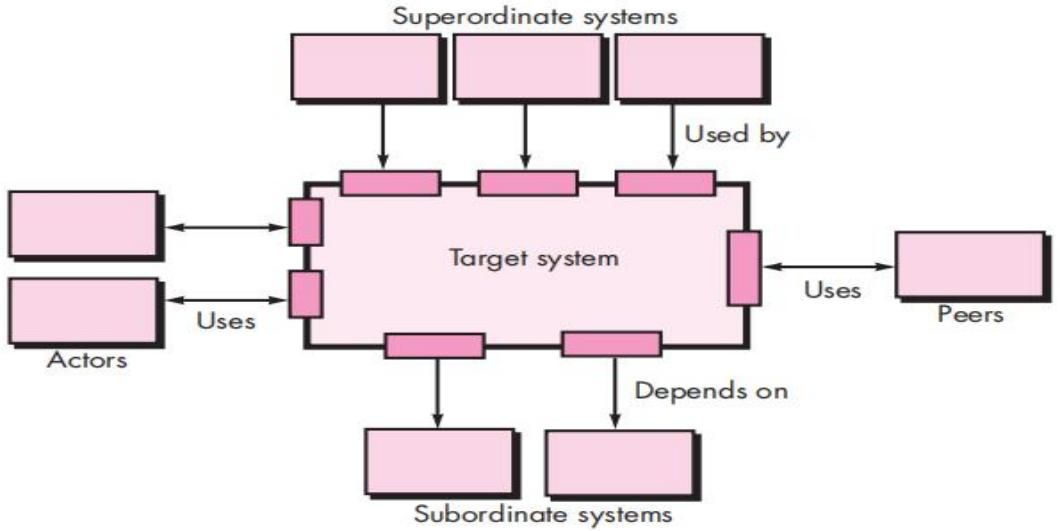
- The software must be placed into context
  - the design should define the external entities (other systems, devices, people) that the software interacts with and the nature of the interaction
- A set of architectural archetypes should be identified
  - An *archetype* is an abstraction (similar to a class) that represents one element of system behavior
- The designer specifies the structure of the system by defining and refining software components that implement each archetype

13.6 Architectural Design

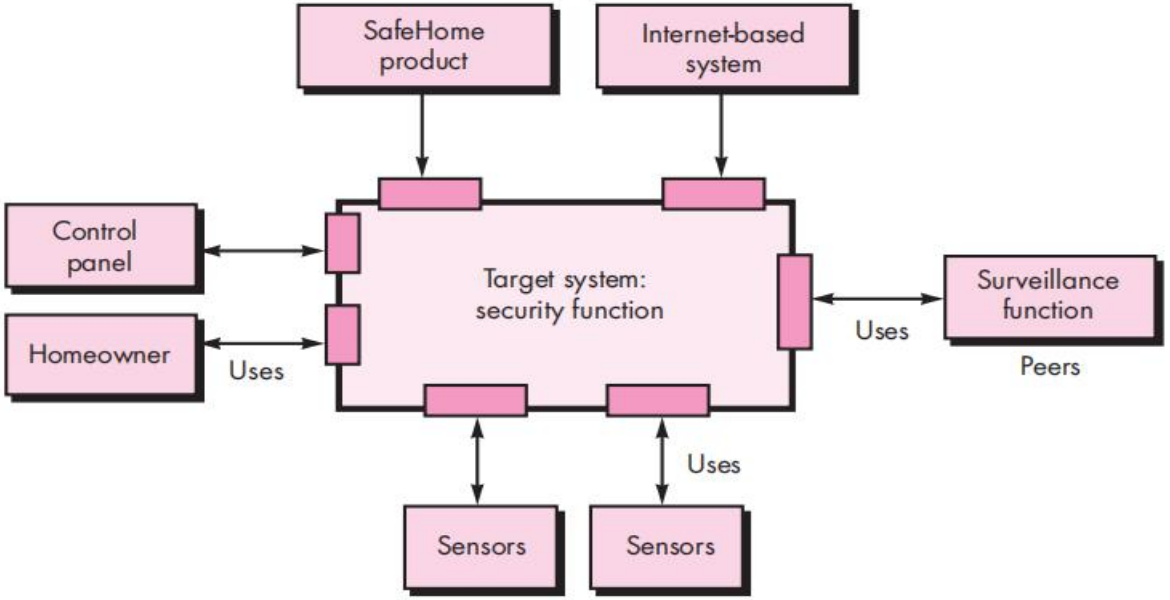
- Useful checklist for Architectural Design
  - Does the diagram show how the system responds to inputs or events?
  - What visualizations might there be to help emphasize areas of risk?
  - How can hidden system design patterns be made more obvious to other developers?
  - Can multiple viewpoints show the best way to refactor specific parts of the system?
  - Can design trade-offs be represented in a meaningful way?

13.6 Architectural Context

ACD
(Architechtural
Context diagaram)

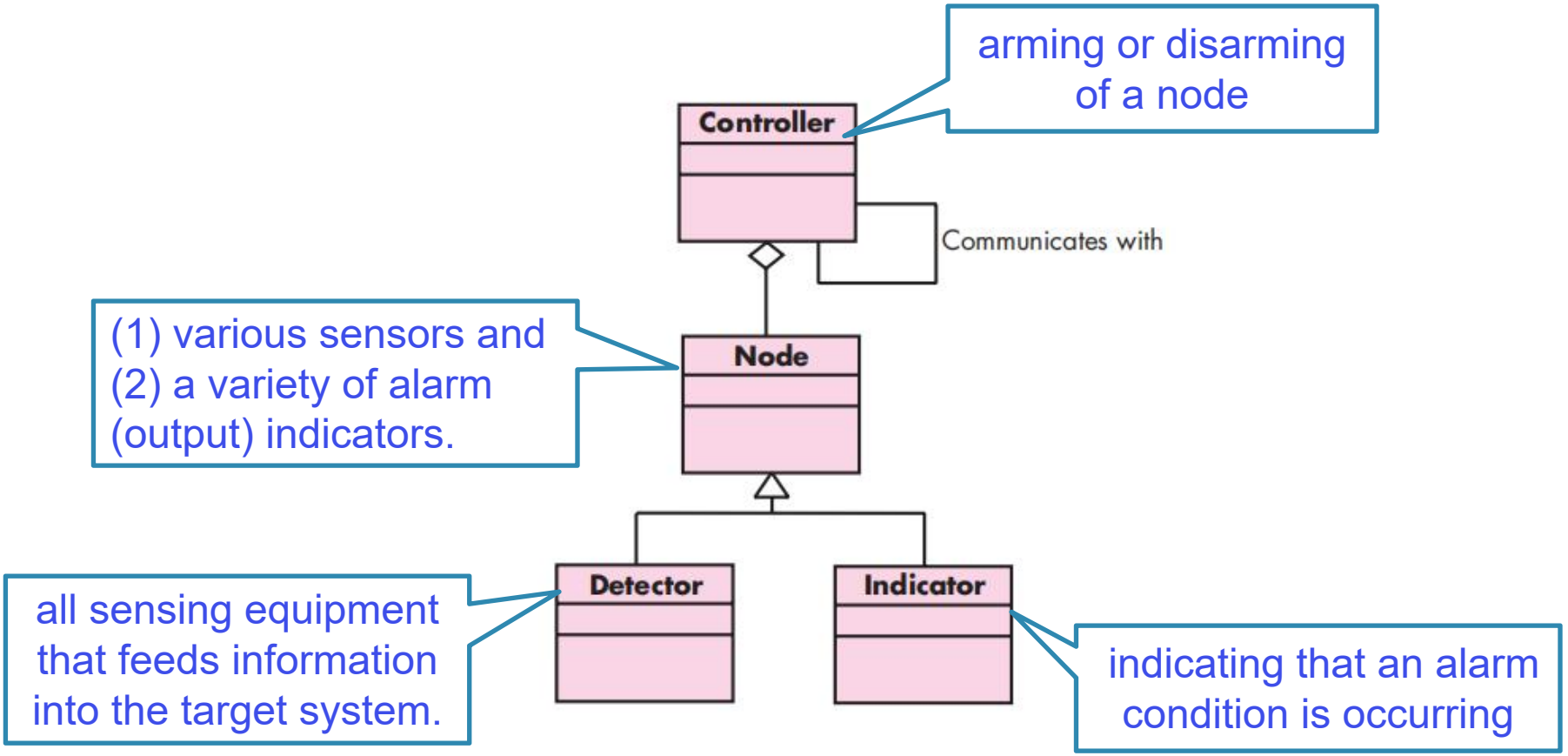


Example



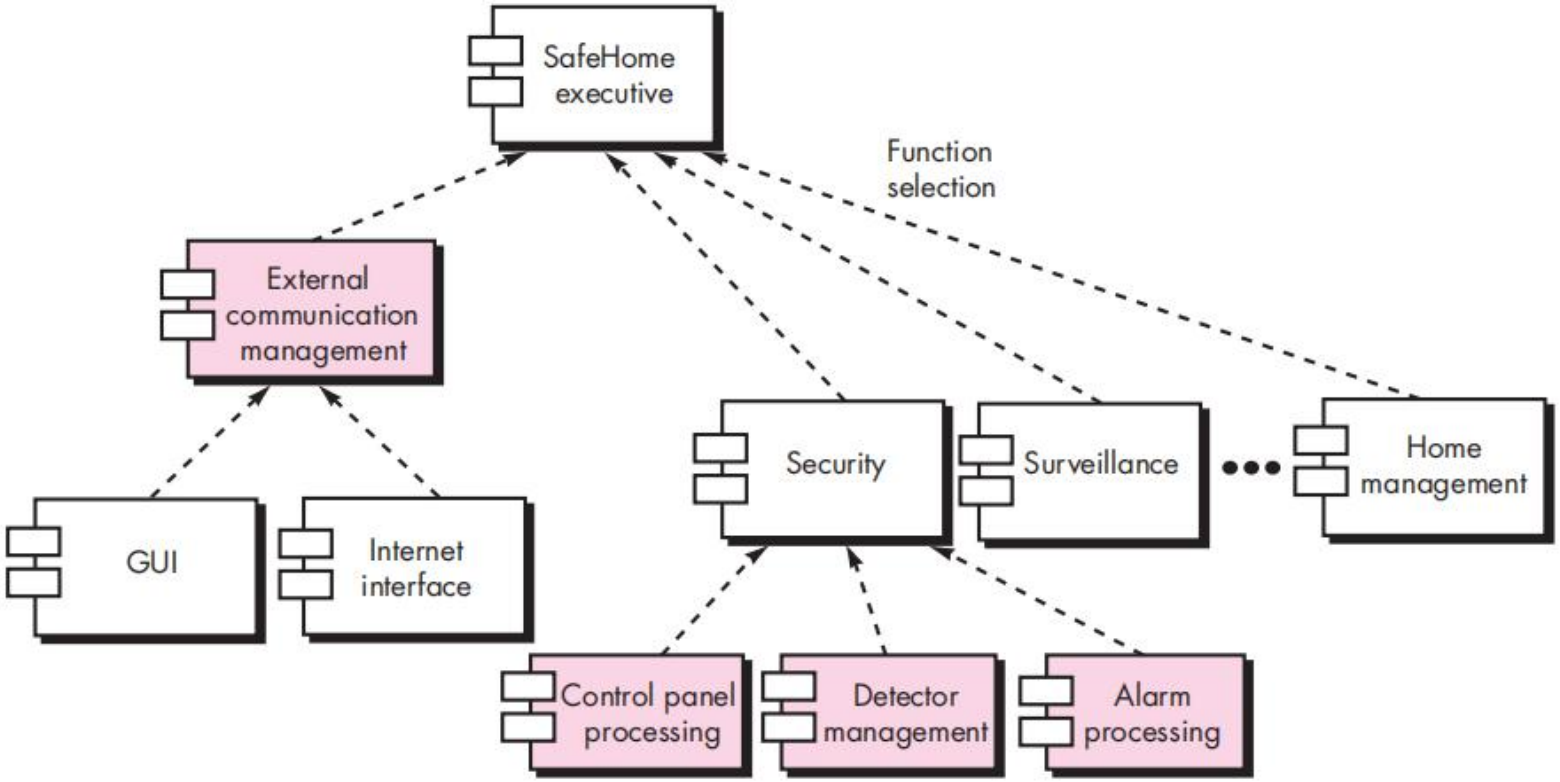
13.6 Archetypes

Define Archetypes (abstraction core block, derived from analysis classes)

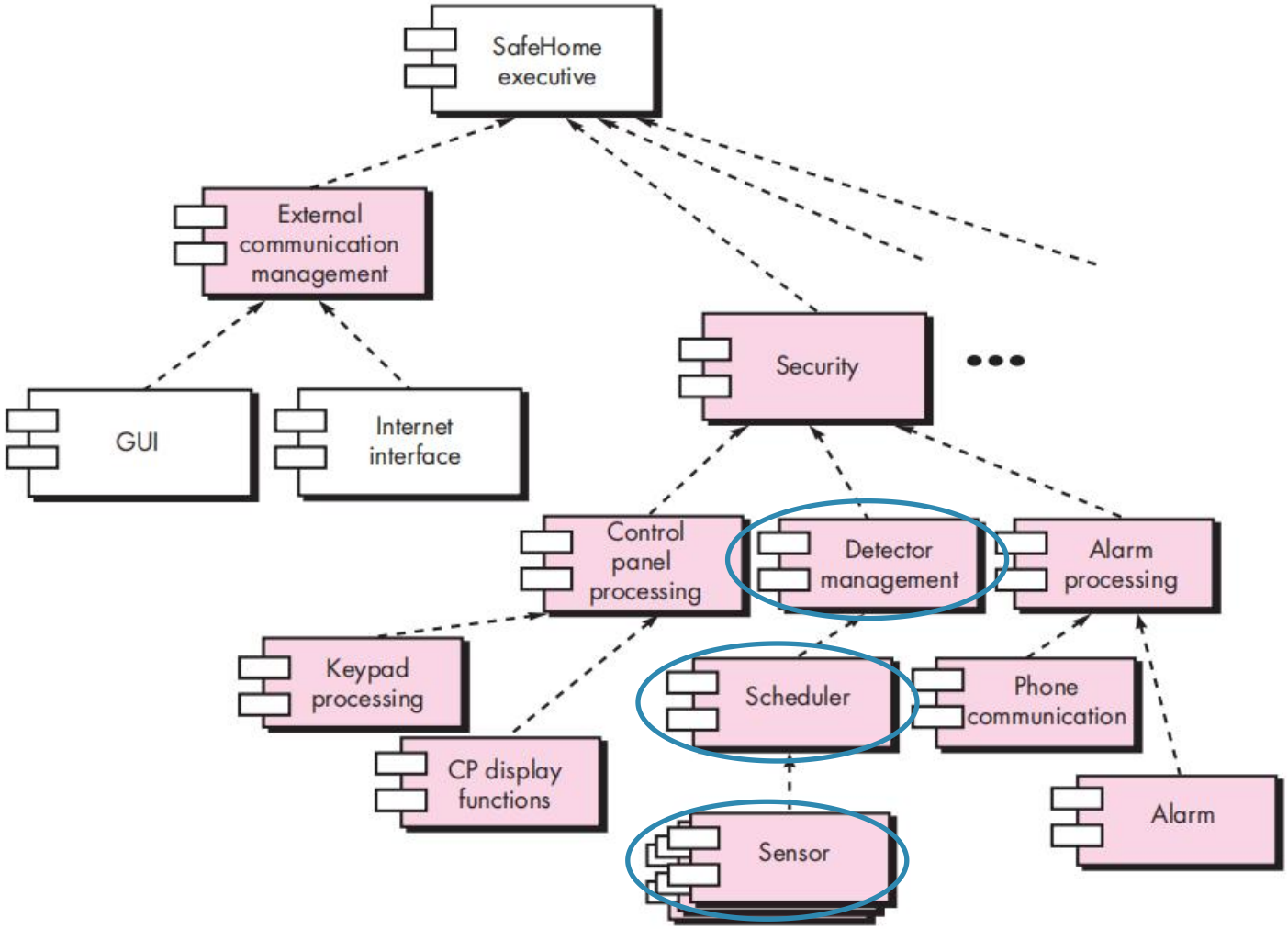


13.6.3 Refined Component Structure

From requirement analysis class or call-return relationship



13.6.4 Example



13.7 Architectural Tradeoff Analysis

An iterative evaluation process for software architectures:

1. Collect scenarios.
2. Elicit requirements, constraints, and environment description.
3. Describe the architectural styles/patterns that have been chosen to address the scenarios and requirements:
   - module view
   - process view
   - data flow view
4. Evaluate quality attributes by considered each attribute in isolation.
5. Identify the sensitivity of quality attributes to various architectural attributes for a specific architectural style.
6. Critique candidate architectures (developed in step 3) using the sensitivity analysis conducted in step 5.

13.7 Architectural Tradeoff Analysis

SAFEHOME

Architecture Assessment

The scene: Doug Miller's office as architectural design modeling proceeds.

The players: Vinod, Jamie, and Ed—members of the *SafeHome* software engineering team and Doug Miller, manager of the software engineering group.

The conversation:

Doug: I know you guys are deriving a couple of different architectures for the *SafeHome* product, and that's a good thing. I guess my question is, how are we going to choose the one that's best?

Ed: I'm working on a call and return style and then either Jamie or I are going to derive an OO architecture.

Doug: Okay, and how do we choose?

Jamie: I took a CS course in design in my senior year, and I remember that there are a number of ways to do it.

Vinod: There are, but they're a bit academic. Look, I think we can do our assessment and choose the right one using use cases and scenarios.

Doug: Isn't that the same thing?

Vinod: Not when you're talking about architectural assessment. We already have a complete set of use cases. So we apply each to both architectures and see how the system reacts, how components and connectors work in the use case context.

Ed: That's a good idea. Makes sure we didn't leave anything out.

Vinod: True, but it also tells us whether the architectural design is convoluted, whether the system has to twist itself into a pretzel to get the job done.

Jamie: Scenarios aren't just another name for use cases.

Vinod: No, in this case a scenario implies something different.

Doug: You're talking about a quality scenario or a change scenario, right?

Vinod: Yes. What we do is go back to the stakeholders and ask them how *SafeHome* is likely to change over the next, say, three years. You know, new versions, features, that sort of thing. We build a set of change scenarios. We also develop a set of quality scenarios that define the attributes we'd like to see in the software architecture.

Jamie: And we apply them to the alternatives.

Vinod: Exactly. The style that handles the use cases and scenarios best is the one we choose.

13.7 Architecture Design Review

- Design review is an essential part of engineering practice

- SAD quality is evaluated in two ways:
  - **Validation**: making sure the design satisfies all of the customer's requirements (i.e., is this the right system?)
  - **Verification**: ensuring the design adheres to good design principles (i.e., are we building the system right?)

13.7 Architecture Design Review

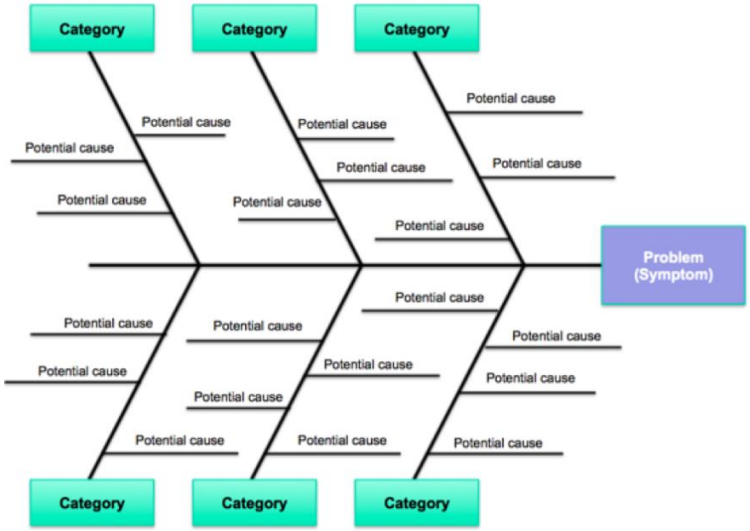
- **Validation**
- Several key people included in review:
  - The analyst(s) who helped define the system requirements
  - The system architect(s)
  - The program designer(s) for this project
  - A system tester
  - A system maintainer
  - A moderator
  - A recorder
  - Other interested developers not otherwise involved in this project

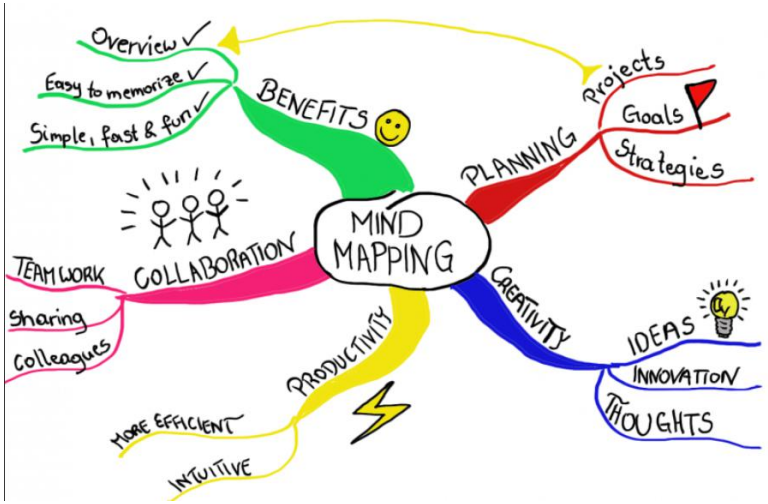
13.7 Architecture Design Review

- **Verification**
- Judge whether it adheres to good design principles:
  - Is the architecture modular, well structured, and easy to understand?
  - Can we improve the structure and understandability of the architecture?
  - Is the architecture portable to other platforms?
  - Are aspects of the architecture reusable?
  - Does the architecture support ease of testing?
  - Does the architecture maximize performance, where appropriate?
  - Does the architecture incorporate appropriate techniques for handling faults and preventing failures?
  - Can the architecture accommodate all of the expected design changes and extensions that have been documented?

13.8 Lesson learned

- Achieve consensus (architects, senior manager, project manager, software engineers)
- Decison analysis and resolution (DAR) methods
  - Chain of causes (root cause analysis)
  - Ishikawa fishbone
  - Mind mapping or spider diagrams





13.9 Pattern-Based Architecture Review

PBAR：an evaluation method that leverages the relationship between architectural patterns and software quality attributes.

1. Identify and discuss the quality attributes by walking through the use cases.
2. Discuss a diagram of system's architecture in relation to its requirements.
3. Identify the architecture patterns used and match the system's structure to the patterns' structure.
4. Use existing documentation and use cases to determine each pattern's effect on quality attributes.
5. Identify all quality issues raised by architecture patterns used in the design.
6. Develop a short summary of issues uncovered during the meeting and make revisions to the walking skeleton.

13.11 Agility and Architecture

- To avoid rework, user stories are used to create and evolve an architectural model (walking skeleton) before coding
- Hybrid models which allow software architects contributing user stories to the evolving storyboard
- Well-run agile projects include delivery of work products during each sprint
- Reviewing code emerging from the sprint can be a useful form of architectural review

Summary

- Architectural Styles
  - Data-centered architectures, Data flow architectures, call and return architectures, Object-oriented architectures, Layered architectures
- How to do architecture design
  - architectural considerations: Economy , Visibility , Spacing , Symmetry , Emergence
  - context and archetypes
  - review / analysis

THE END