
LUT
University

Software Engineering Models and Modeling

Software Architecture and Modelling Architecture

Antti Knutas

Introduction to Software Architecture

Consider this: What is ”Architecture”?

- ”Architecture” is a common word
- Everyone use the concept ”architecture”
- Define, what you mean about the concept ”architecture”, when you speak about software and information systems
  - For example
    - What is included in architecture?
    - What are the uses of architecture?
- Try to be as concrete as possible and use examples

From the dictionary

Main Entry: **ar·chi·tec·ture**
Function: *noun*
1 : the art or science of building; *specifically* **:** the art or practice of designing and building structures and especially habitable ones
2 a : formation or construction as or as if as the result of conscious act <the *architecture* of the garden> **b :** a unifying or coherent form or structure <the novel lacks *architecture*>
3 : architectural product or work
4 : a method or style of building
5 : the manner in which the components of a computer or computer system are organized and integrated

From Merriam-Webster's Collegiate Dictionary, 10th edition (Merriam-Webster 2002)

Architecture in Software Engineering

- Architecture is a metaphor
  - Metaphor: understanding an abstract phenomenon in terms of another, more concrete phenomenon
- Understanding a virtual machine or abstract construction in terms of a physical building
- Building architecture as a metaphor

Formal definition

"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" (Bass et al., 1998, p. 23).

- According to the definition, architecture consists of
  - Structures that are composed of components
  - Externally visible properties of the components
  - The relationships between the component
- Open questions
  - What are these structures? Why are they "architecture"?
  - => there are many possible ways to view software architecture
  - We'll get back to this in 4+1 views to architecture

Why is architecture needed?

Computer and software systems are becoming increasingly complex and interconnected: Increase distribution, integration, and in diversity of interfaces.

- 1980s: Client-server, databases
- 1990s: Component architectures, Internet, mobile, web services
- 2000s: The cloud, distributed systems
- 2010s: Microservices, blockchains

How architecture is believed to help in all this?

- Management of complexity
  - Understanding technology and solutions
  - Stakeholder communication (focus of discussion by system stakeholders)
- Large-scale reuse
  - The architecture may be reusable across a range of systems
  - Product-line architectures may be developed.
- System analysis and communication enabling better quality
  - Management of non-functional requirements, like performance, maintainability, security, …
  - Architecture has the most important role in meeting the non-functional requirements

Connection of architecture to software engineering processes (examples)

- Specification of the system structure
  - Architecture is the basis for further design and implementation
- Basis for work division
  - Architecture may even define the organizational structure
- Basis for reuse
  - Also the use of third party products is specified by architecture
- Management of technological risks
  - Selection of tools and techniques
- Support of strategic decisions
- Creation of common understanding between stakeholders

Architectural Design

Design decisions and what to consider when creating architecture (and architecture models)

Intro to Architectural Design

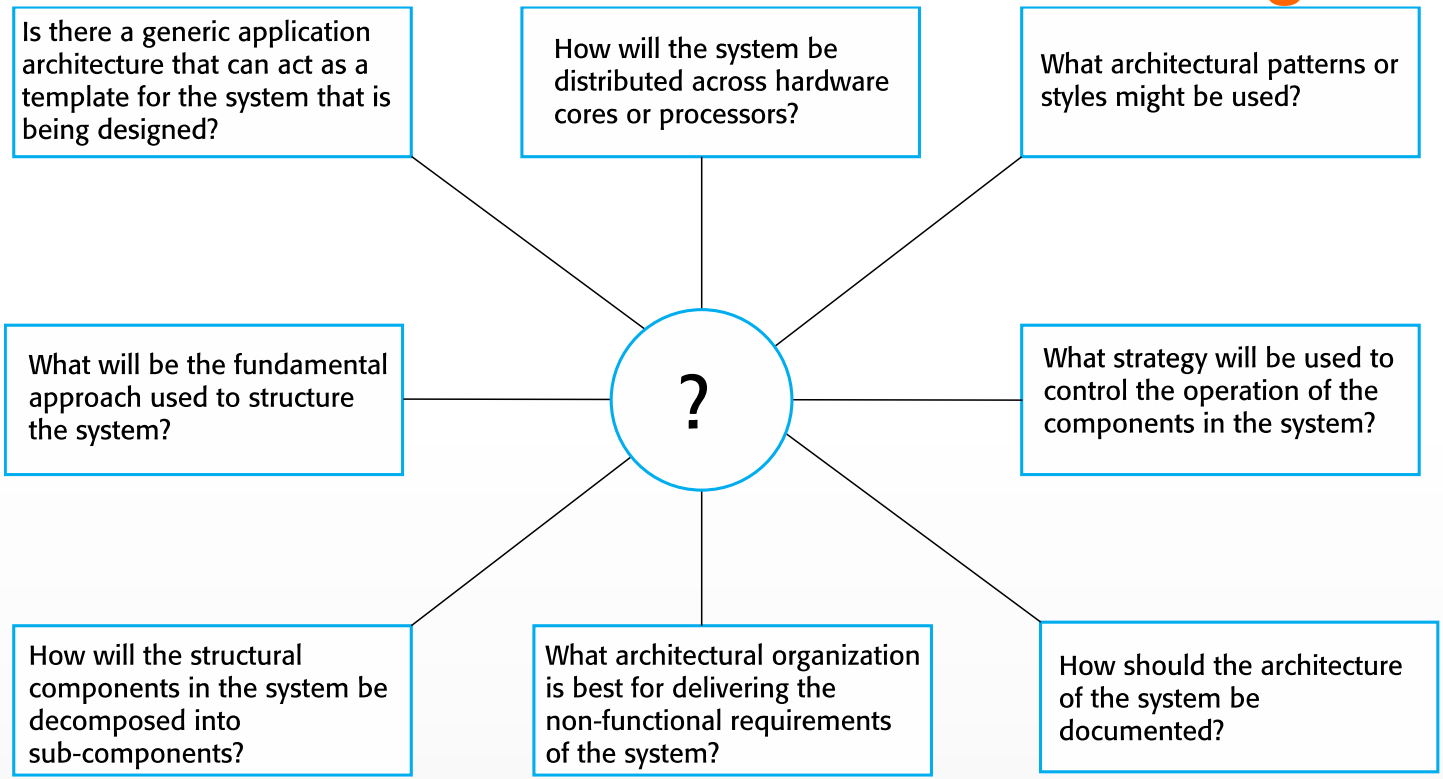
- Architectural design is concerned with understanding how a software system should be organized and designing the overall structure of that system.
- Architectural design is the critical link between design and requirements engineering, as it identifies the main structural components in a system and the relationships between them.
- The output of the architectural design process is an architectural model that describes how the system is organized as a set of communicating components.
- Even in agile processes, overall architecture is designed at an early stage
  - Refactoring the system architecture is usually expensive, since it affects many components in the system

We will cover the introduction, and more is discussed in a future Software Architecture course available in our master's program.

What is included in architecture design?

- A creative process – depends on the system being developed, available resources, and expertise of designers
- We'll review some typical questions next

What is included in architecture design?



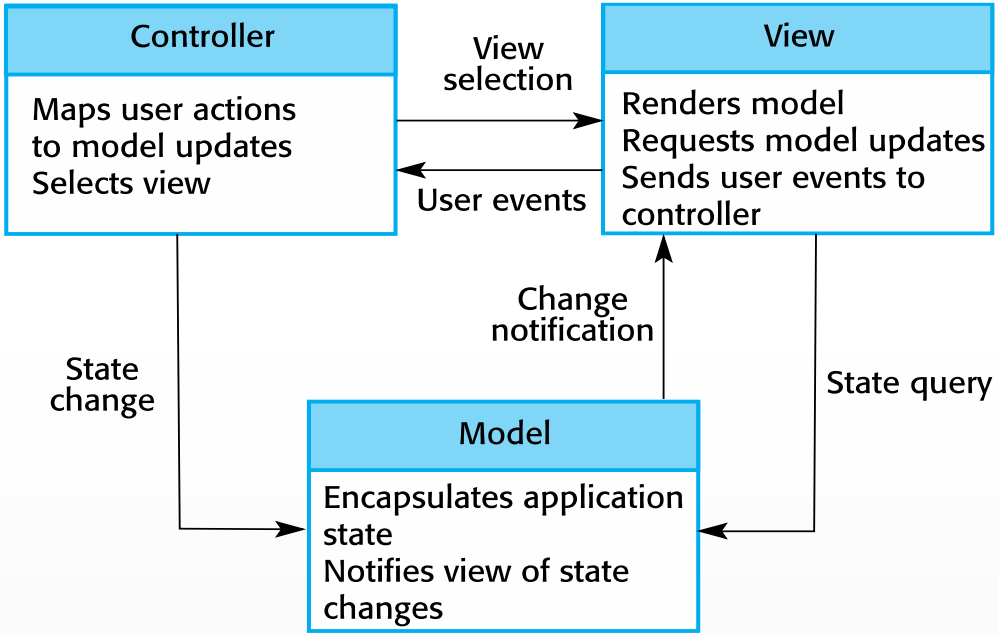
How design decisions can affect system characteristics

- Performance
  - Localise critical operations and minimise communications. Use large rather than fine-grain components.
- Security
  - Use a layered architecture with critical assets in the inner layers.
- Safety
  - Localise 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.

Some example architecture patterns: MVC

Name	MVC (Model-View-Controller)
Description	Separates presentation and interaction from the system data. The system is structured into three logical components that interact with each other. The Model component manages the system data and associated operations on that data. The View component defines and manages how the data is presented to the user. The Controller component manages user interaction (e.g., key presses, mouse clicks, etc.) and passes these interactions to the View and the Model. See Figure 6.3.
Example	Figure 6.4 shows the architecture of a web-based application system organized using the MVC pattern.
When used	Used when there are multiple ways to view and interact with data. Also used when the future requirements for interaction and presentation of data are unknown.
Advantages	Allows the data to change independently of its representation and vice versa. Supports presentation of the same data in different ways with changes made in one representation shown in all of them.
Disadvantages	Can involve additional code and code complexity when the data model and interactions are simple.

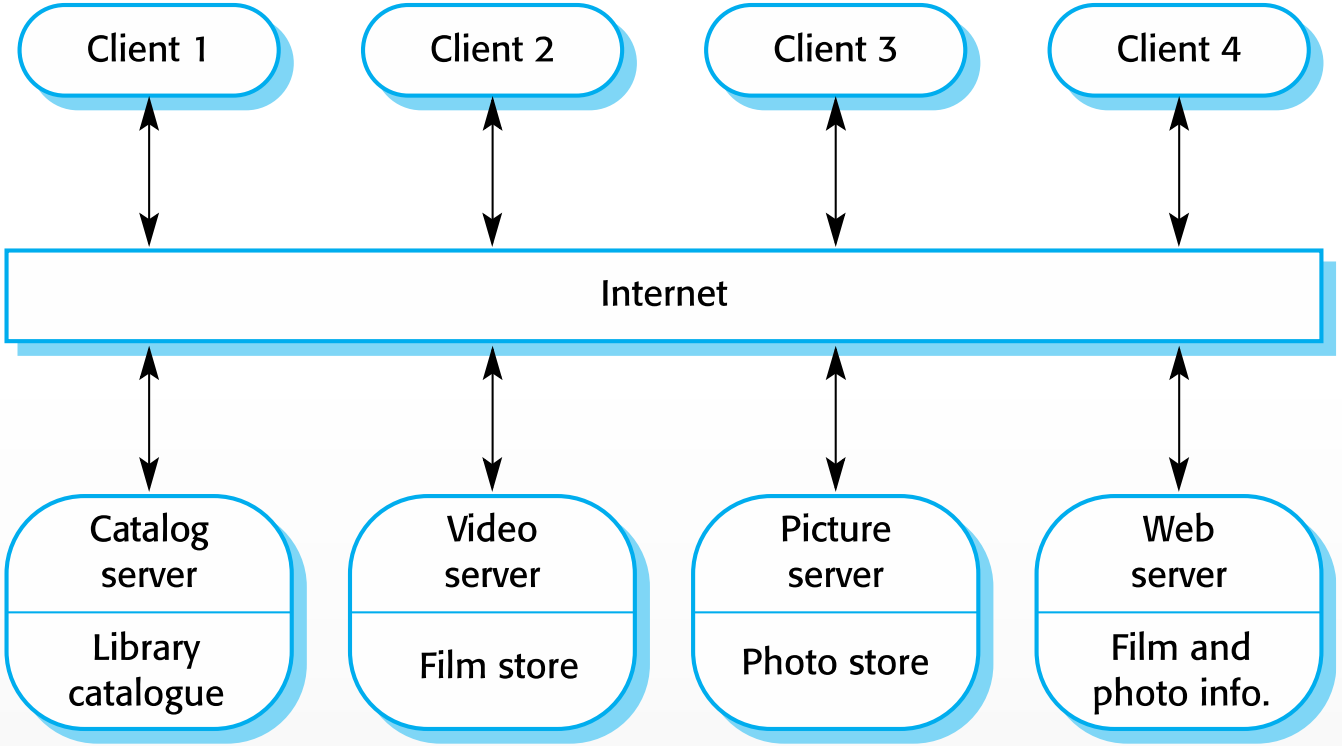
Some example architecture patterns: MVC



Some example architecture patterns: Client - server

Name	Client-server
Description	In a client–server architecture, the functionality of the system 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.
Example	Figure 6.11 is an example of a film and video/DVD library organized as a client–server system.
When used	Used when data in a shared database has to be accessed from a range of locations. Because servers can be replicated, may also be used when the load on a system is variable.
Advantages	The principal advantage of this model is that servers can be distributed across a network. General functionality (e.g., a printing service) can be available to all clients and does not need to be implemented by all services.
Disadvantages	Each service is a single point of failure so susceptible to denial of service attacks or server failure. Performance may be unpredictable because it depends on the network as well as the system. May be management problems if servers are owned by different organizations.

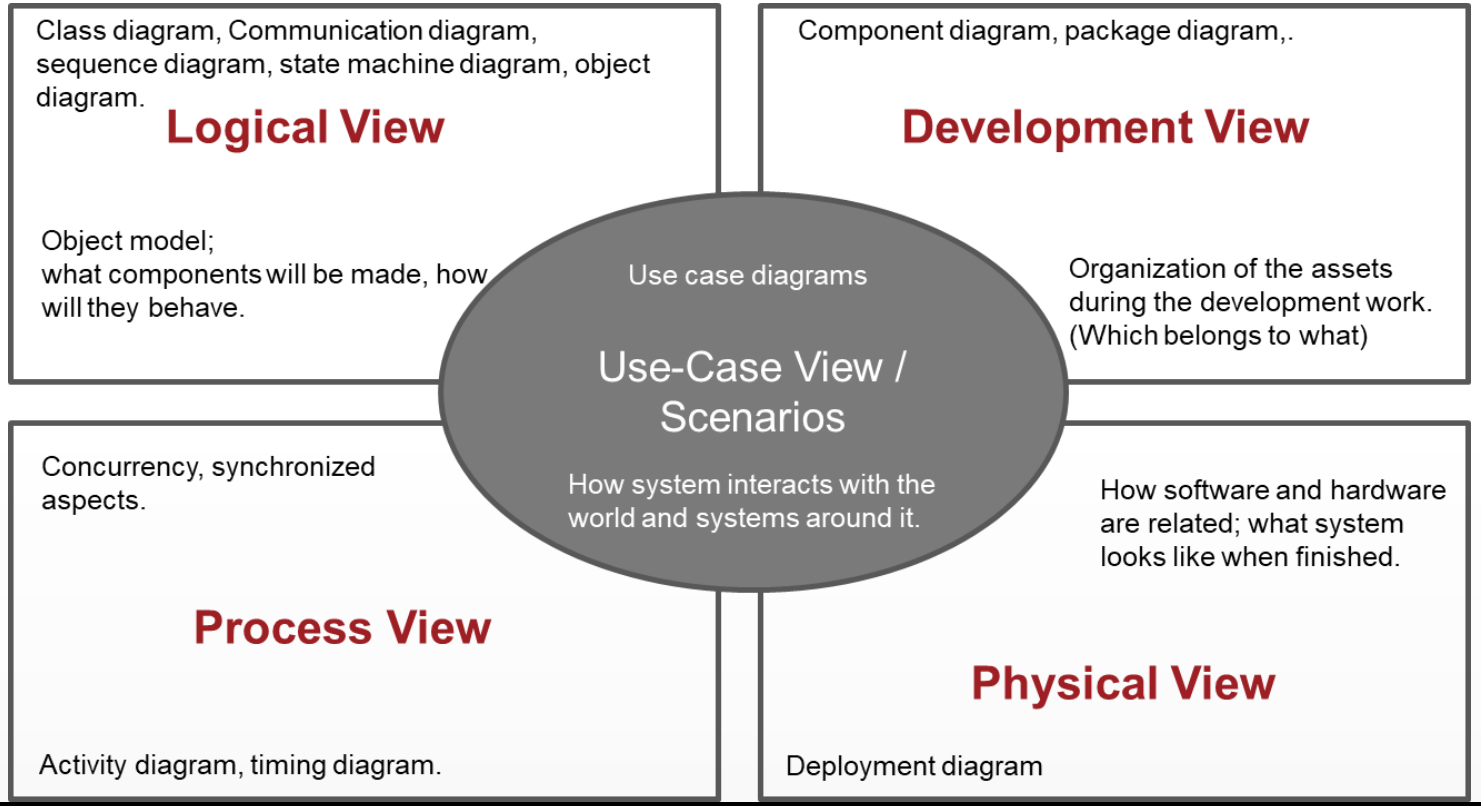
Some example architecture patterns: Client - server



Architectural Views

Different ways to model and view software architecture

4+1 view into architecture



4+1 (continued)

- A logical view,
  - which shows the key abstractions in the system as objects or object classes.
  - Static and dynamic structure.
  - Class, communication, sequence, state machine, object diagrams
- A process view,
  - which shows how, at run-time, the system is composed of interacting processes.
  - Mechanisms of concurrency and processes.
  - Activity, timing diagrams
- A development view,
  - which shows how the software is decomposed for development.
  - Structure of source code and software components.
  - Component, package diagrams
- A physical view,
  - which shows the system hardware and how software components are distributed across the processors in the system.
  - Deployment diagram
- Related use cases or scenarios (+1)
  - Functional requirements

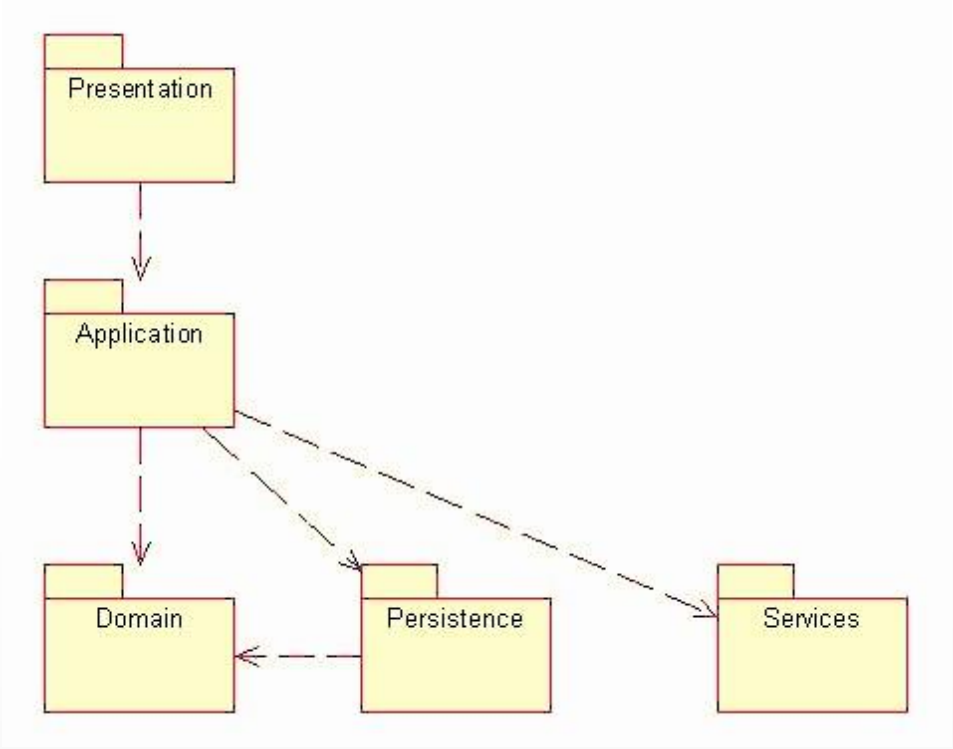
”Software Architecture Document” sample template from RUP

1. Introduction
   - 1.1 Purpose
   - 1.2 Scope
   - 1.3 Definitions, Acronyms and Abbreviations
   - 1.4 References
   - 1.5 Overview
2. Architectural Representation
3. Architectural Goals and Constraints
4. Use-Case View
5. Logical View
   - 5.1 Overview
   - 5.2 Architecturally Significant Design Packages
   - 5.3 Use-Case Realizations
6. Process View
7. Deployment View
8. Implementation View
   - 8.1 Overview
   - 8.2 Layers
9. Data View (optional)
10. Size and Performance
11. Quality

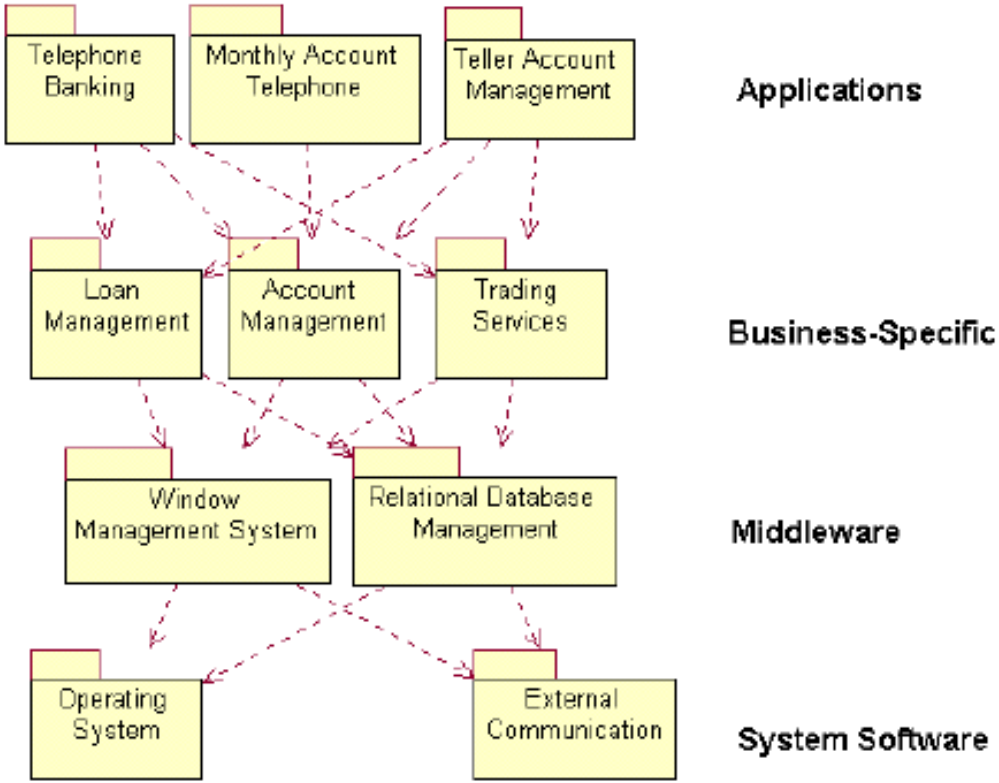
Architecture diagram examples

In UML

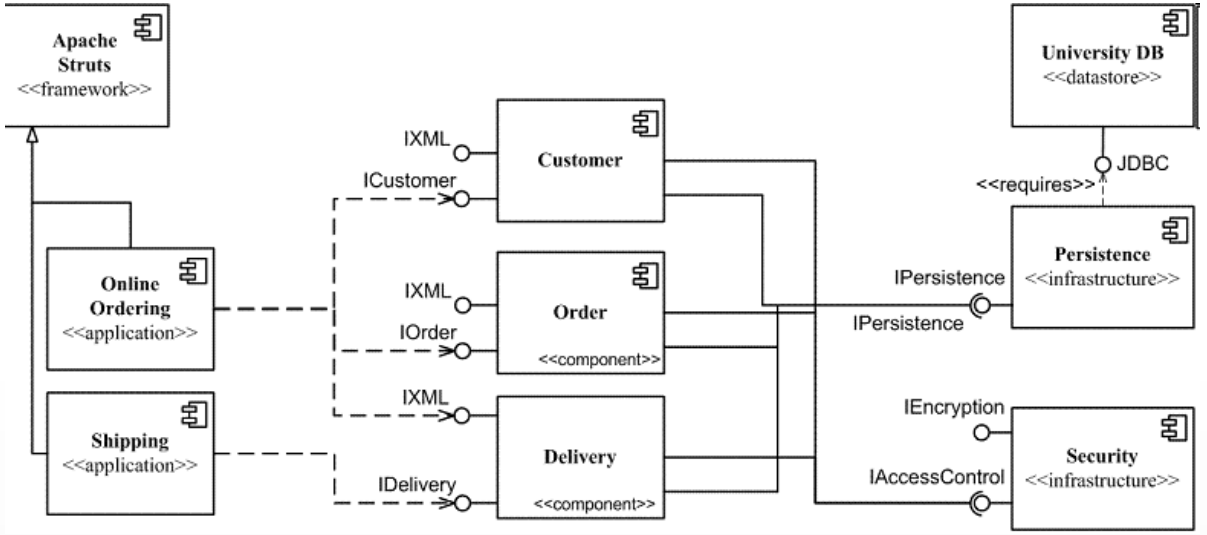
Subsystem dependencies presented with packages



Layer thinking with packages



Component diagrams



Extended notation from 1.x

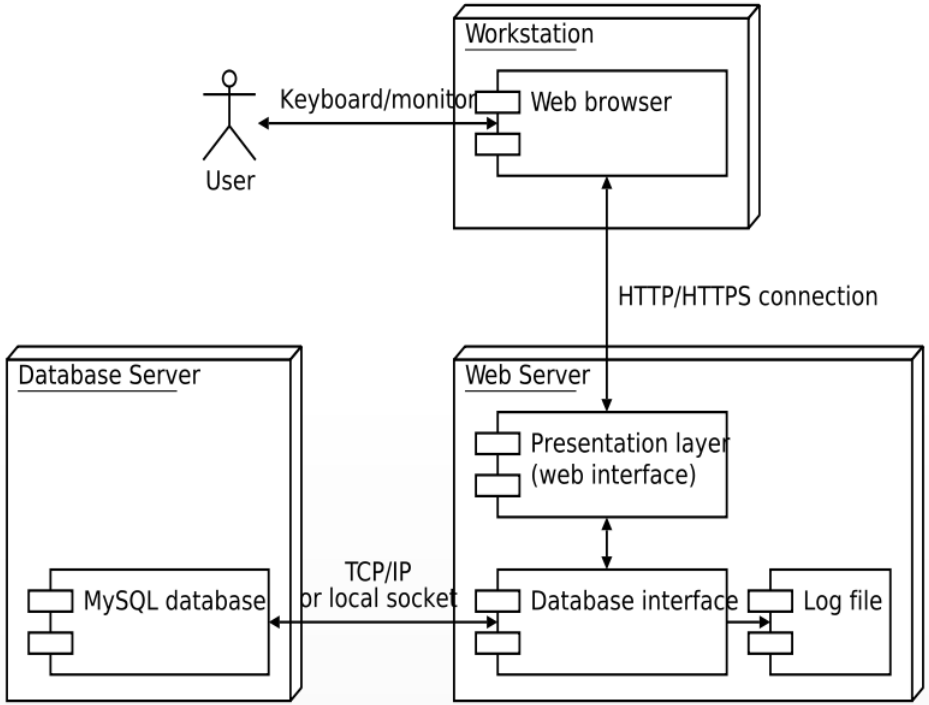
- Out/In Ports with interfaces
- Change in component symbol

An application level “wiring diagram”

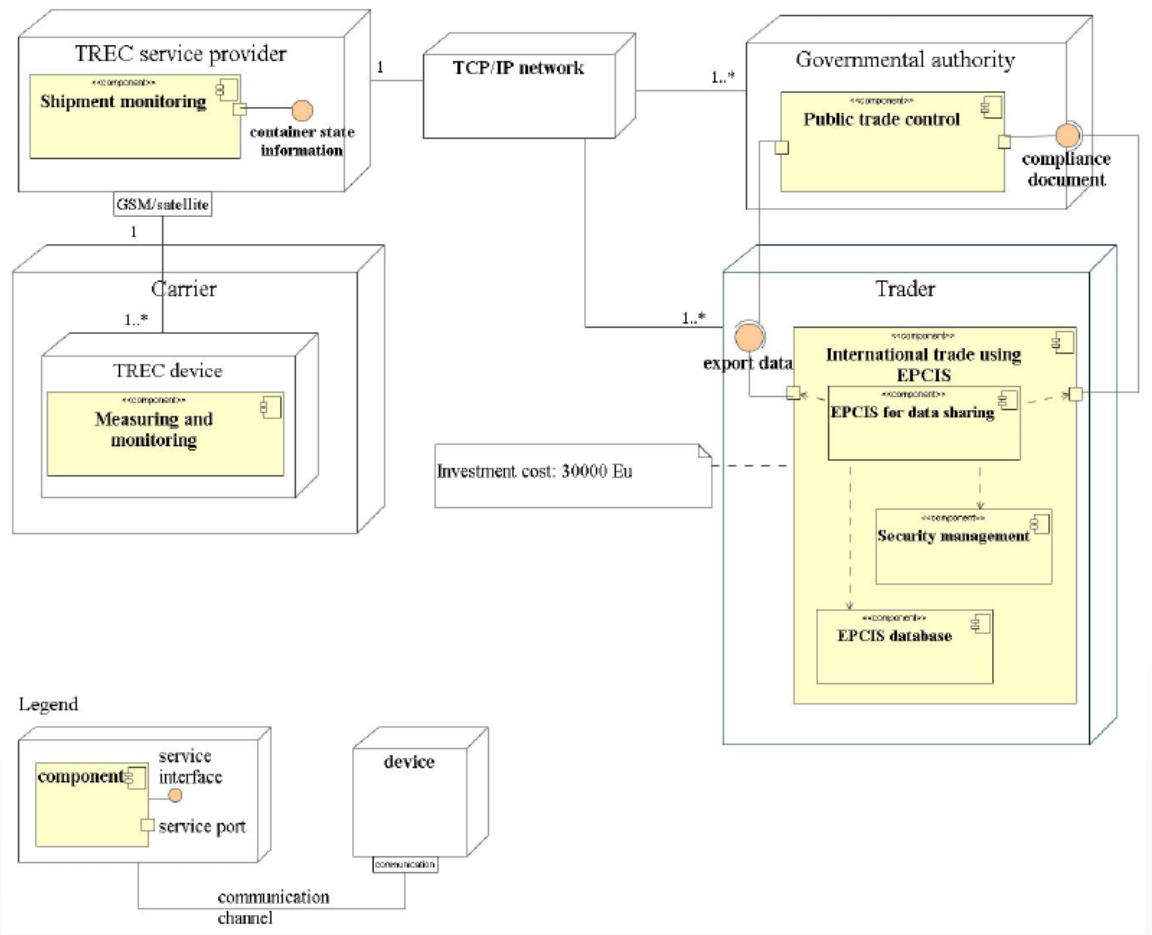
Deployment diagram

External symbol belong to hardware

Internal symbols present software components in the runtime environment

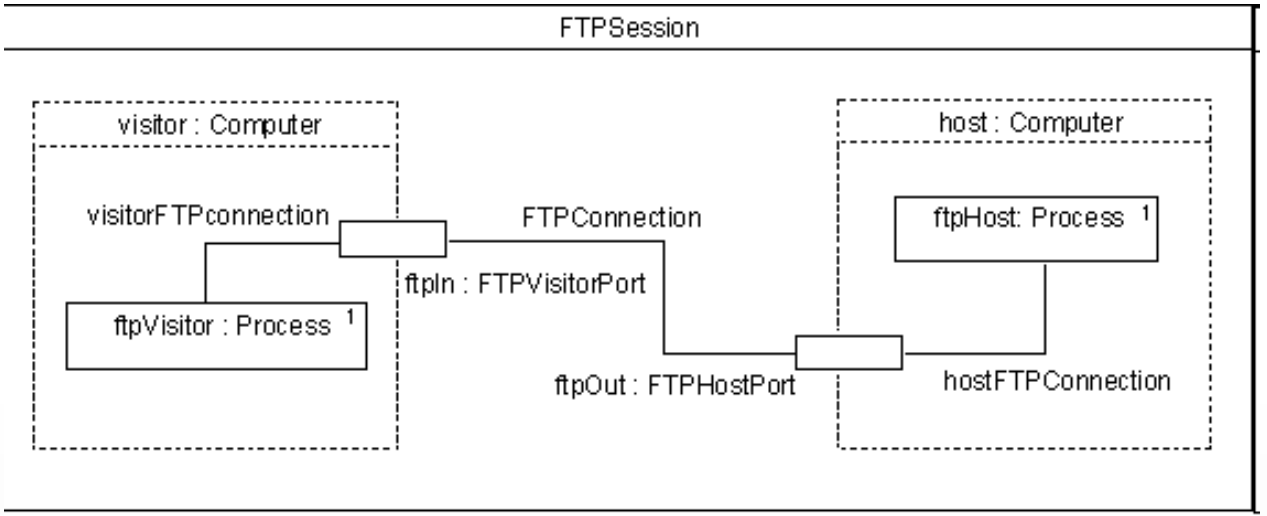


Deployment diagram



Kraussl, Z. & Tan, Yao-Hua & Gordijn, Jaap. (2009). A Model-Based Approach to Aid the Development of E-Government Projects in Real-Life Setting Focusing on Stakeholder Value. Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, HICSS. 1 - 10. 10.1109/HICSS.2009.24.

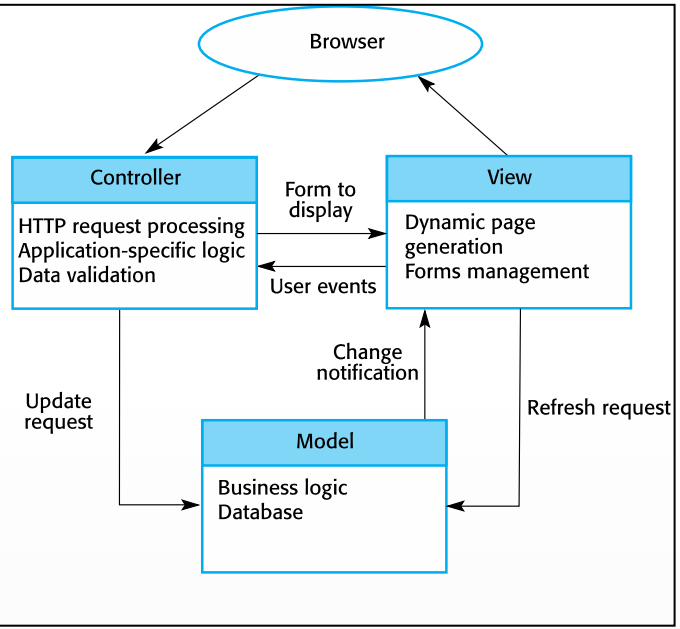
Composite structure diagram

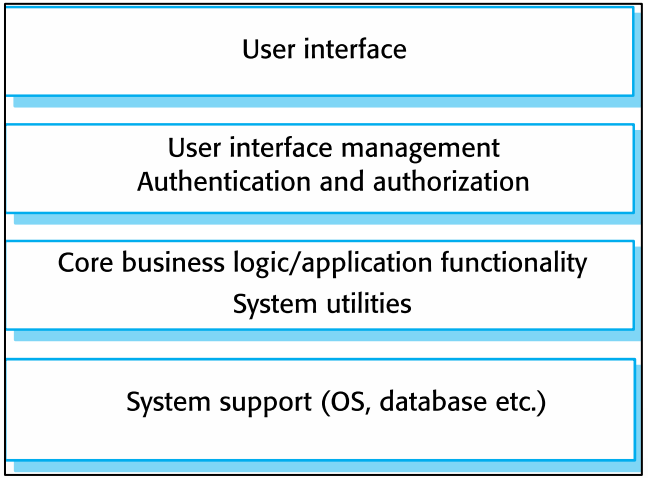


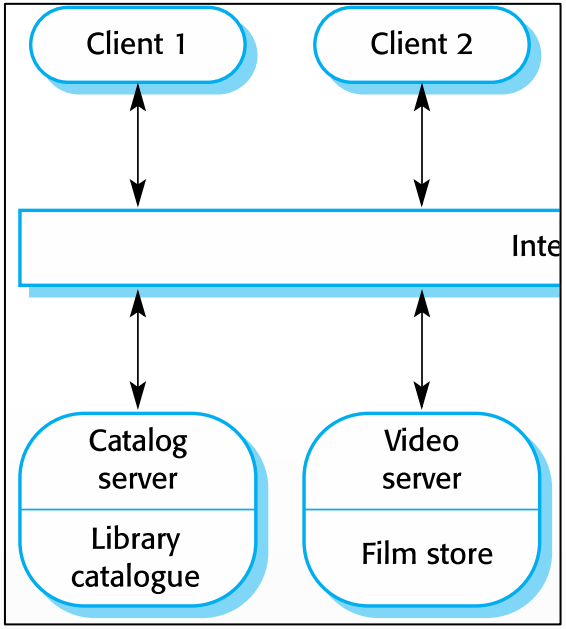
Enables modelling of hierarchical object structures

- Ports, connectors, message routes

Free-form examples (MVC, layer, client-server)







You will use free-form notations like these most likely in your working life, but course assignments require UML.

Summary

Key points

- A software architecture is a description of how a software system is organized.
- Architectural design decisions include decisions on the type of application, the distribution of the system, the architectural styles to be used.
- Architectures may be documented from several different perspectives or views such as a conceptual view, a logical view, a process view, and a development view.
- Architectural patterns are a means of reusing knowledge about generic system architectures. They describe the architecture, explain when it may be used and describe its advantages and disadvantages.
- Models of application systems architectures help us understand and compare applications, validate application system designs and assess large-scale components for reuse.

Key points

- As a concept, architecture is not always well-defined
  - People mean many things when they speak about architecture
  - Software architecture and system architecture are the most common terms
- In spite of this inclarity (or perhaps because of), architecture is an important tool through a software engineering process, including
  - Determining and fulfilling non-functional requirements
  - Managing the complexity
  - Work division
  - Disseminating knowledge and information
  - Making decisions about technology