



Chapter Four

Software Design

Department of Software Engineering
WCU

9 April 2024

Software design

- Software design is a meaningful representation of something to be built.
- Software design is a mechanism to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.
- It deals with representing the client's requirement, as described in SRS (Software Requirement Specification) document, into a form, i.e., easily implementable using programming language.

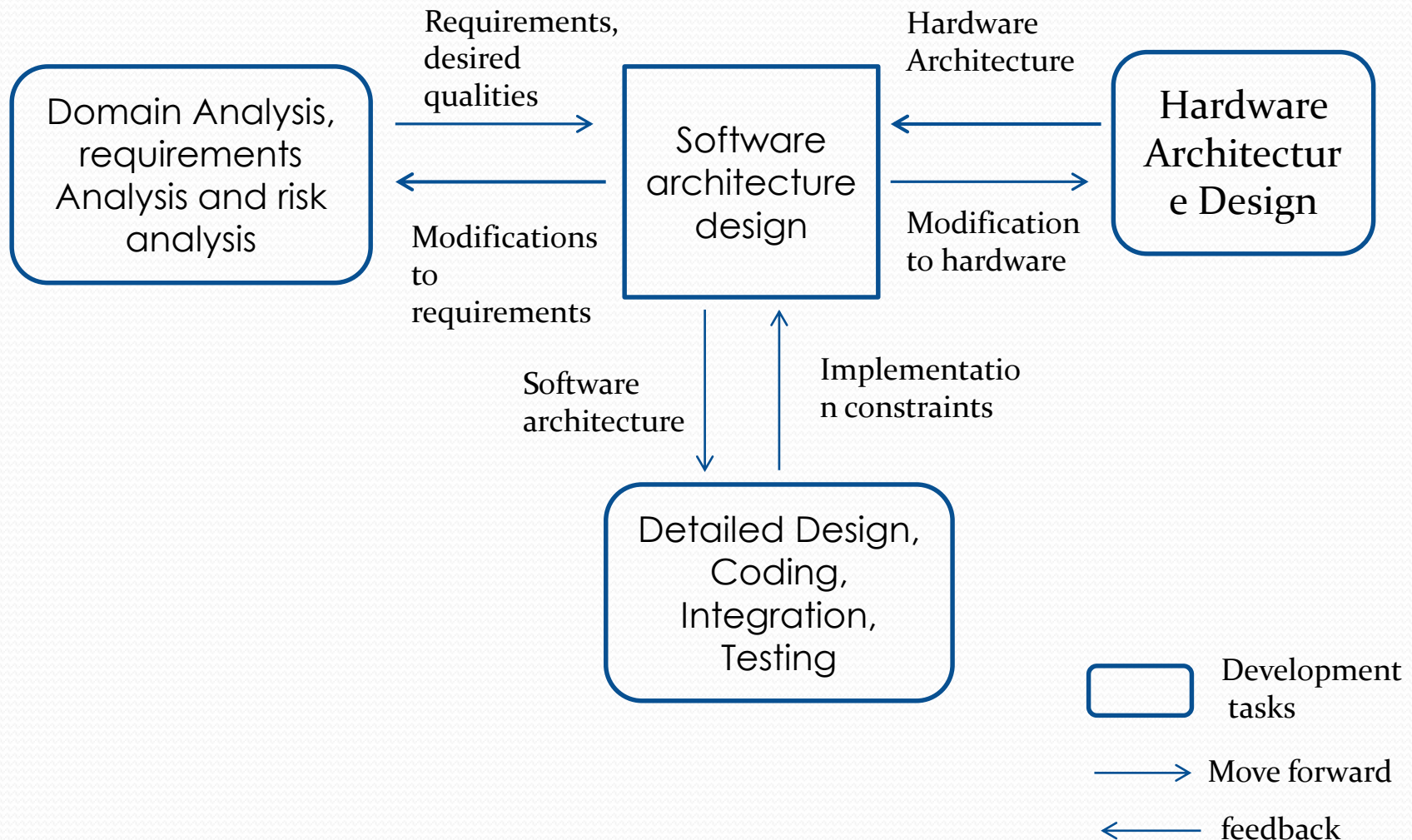
Software Design

- Software design involves a set of **principles**, **concepts** and **practices** that lead to the development of high quality software
- For design, software engineers are looking more at the ***solution domain*** (rather than the problem domain when doing requirements)
- The output is ***models that are used for system implementation***, i.e. **models that are architectural, interface, component-level, Data and deployment representations of the system**

The Design Process

- The design is the *blueprint* of the system to be implemented in order to produce quality software
- The **quality guidelines** used are:
 - The design must implement all of the explicit requirements contained in the analysis model, and it must accommodate all of the implicit requirements desired by the customer.
 - The design must be a readable, understandable guide for those who generate code and for those who test and subsequently support the software.
 - The design should provide a complete picture of the software, addressing the data, functional, and behavioral domains from an implementation perspective.

Relation of Software architecture to other development tasks



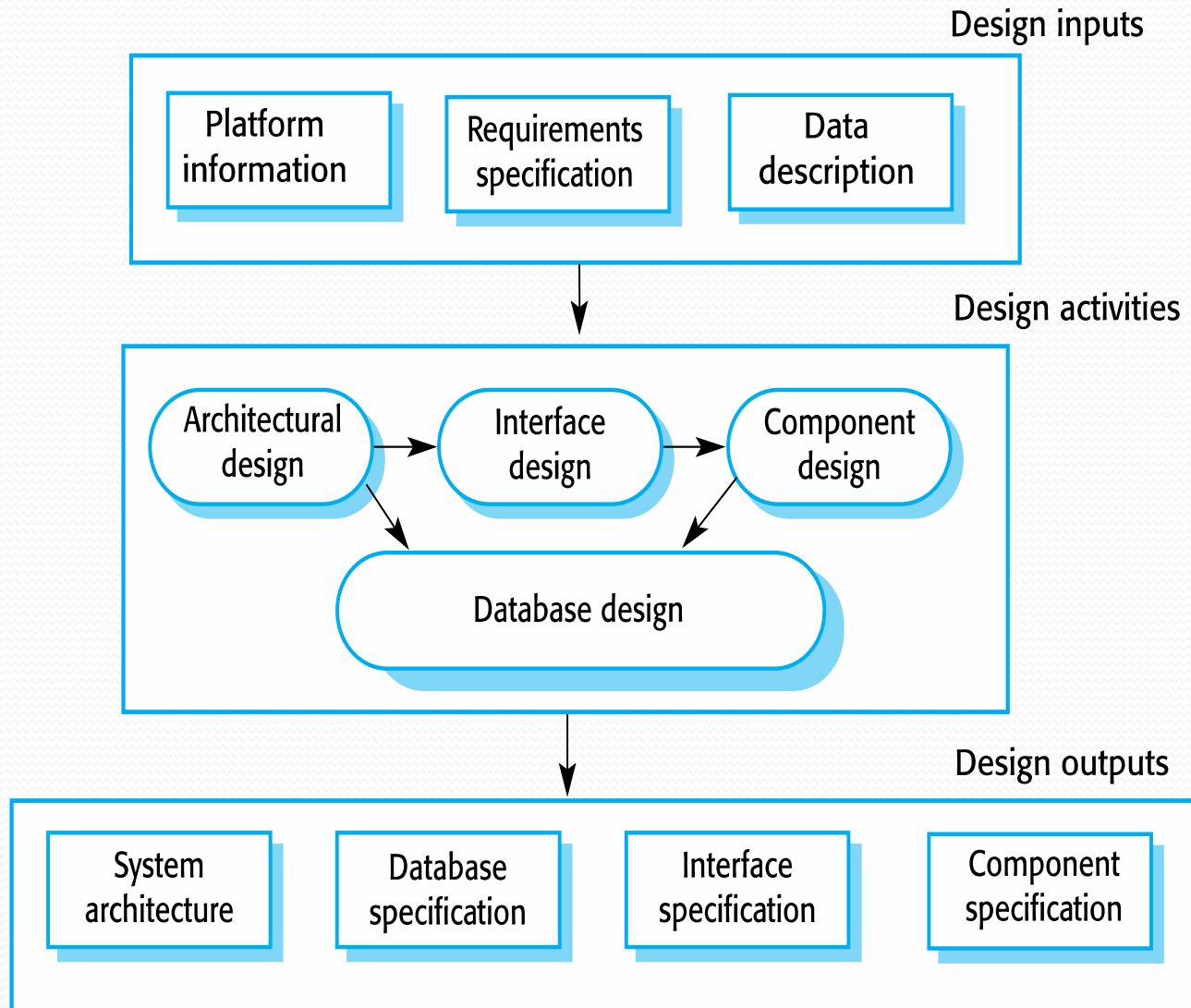
Software Design

- Software design model consists of 4 designs:
 - Data/class Design
 - Architectural Design
 - Interface Design
 - Component Design

Cont....

- **Data/class design** - Created by transforming the analysis model class-based elements into classes and data structures required to implement the software. You get the data structure from entity r/ship at analysis stage.
- **Architectural design** - defines the relationships among the major structural elements of the software, it is derived from the class-based elements and flow-oriented elements of the analysis model
- **Interface design** - describes how the software elements, hardware elements, and end-users communicate with one another, it is derived from the analysis model scenario-based elements, flow-oriented elements, and behavioral elements
- **Component-level design** - created by transforming the structural elements defined by the software architecture into a procedural description of the software components using information obtained from the analysis model class-based elements, flow-oriented elements, and behavioral elements
 - Procedure description of sw in terms of data flow diagram. And how the data is flowing between subsystems.

General model of the design process



Design quality attributes

- Acronym FURPS –
 - Functionality
 - Usability
 - Reliability
 - Performance
 - Supportability

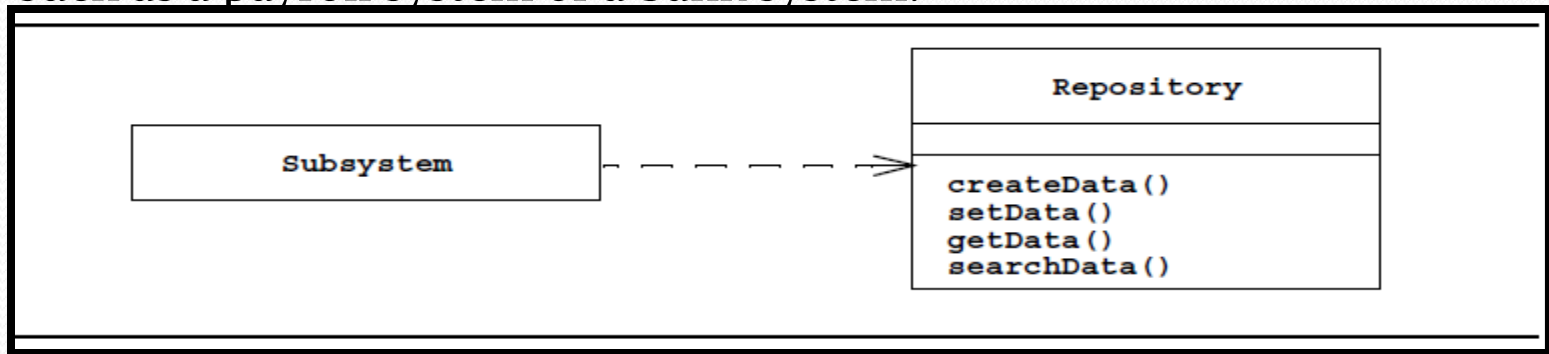
- Functionality – is assessed by evaluating the feature set and capabilities of the program.
 - Functions that are delivered and security of the overall system.
- Usability – assessed by considering human factors, consistency & documentation.
- Reliability – evaluated by
 - measuring the frequency and severity of failure.
 - Accuracy of output results.
 - Ability to recover from failure and predictability of the program.
- Performance - measured by processing speed, response time, resource consumption, efficiency.
- Supportability – combines the ability to extend the program (extensibility), adaptability and serviceability.

Software architecture

- A **software architecture** includes the system decomposition, the global control flow, error-handling policies and inter subsystem communication protocols [Shaw & Garlan, 1996].

Some of architectural type

1. **Repository architecture:** An architecture where subsystems access and modify data from a **single data structure** called the central **repository**.
 - Control flow can be dictated either by the central repository
 - The repository architecture is typical for **database management systems**, such as a payroll system or a bank system.



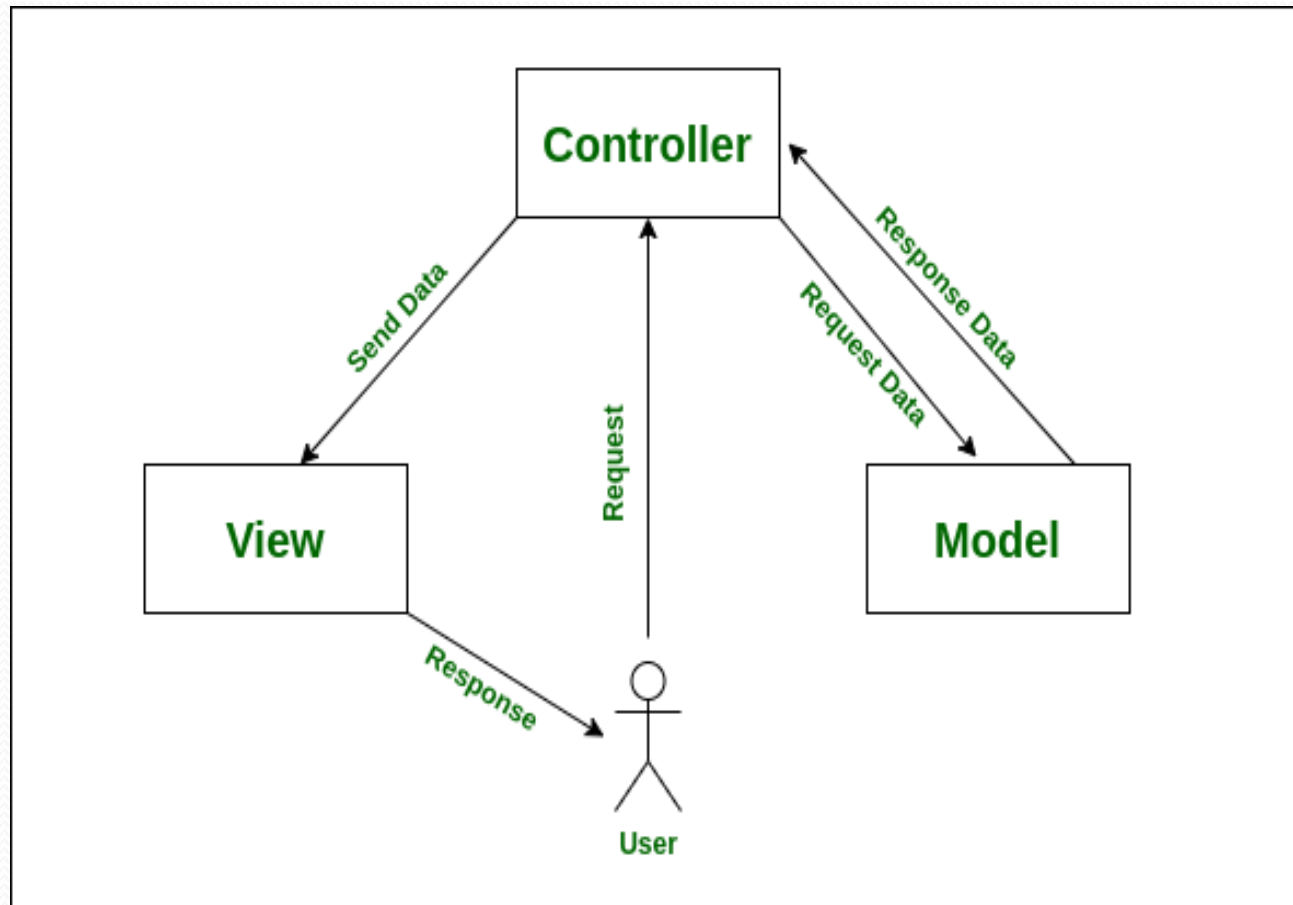
- Fig. Repository architecture (UML class diagram). Every subsystem depends only on a central data structure called the repository. The repository in turn, has no knowledge of the other subsystems.

Cont...

2. *Model/View/Controller:*

- In the Model/View/Controller (MVC) architecture subsystems are classified into three different types:
 1. **model subsystems** are responsible for maintaining domain knowledge,
 - It is developed such that they do not depend on any view or controller subsystem.
 - It maintain the central data structure.
 2. **view subsystems** are responsible for displaying it to the user, and
 3. **Controller subsystems** are responsible for managing the sequence of interactions with the user. It gather input from the user and send message to the model.

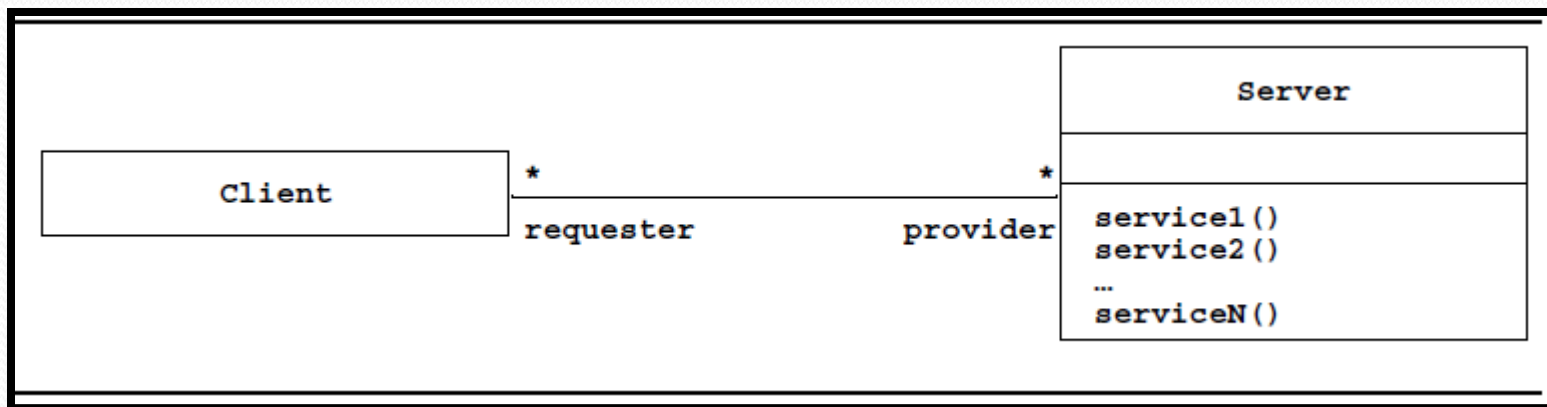
MVC



Cont...

3. *Client/server architecture*

- In the client/server architecture a subsystem, the **server**, provides services to instances of other subsystems called the **clients**, which are responsible for interacting with the user.
- The request for a service is usually done via a remote procedure call mechanism or a common object broker.
- The client/server architecture is a generalization of the repository architecture.

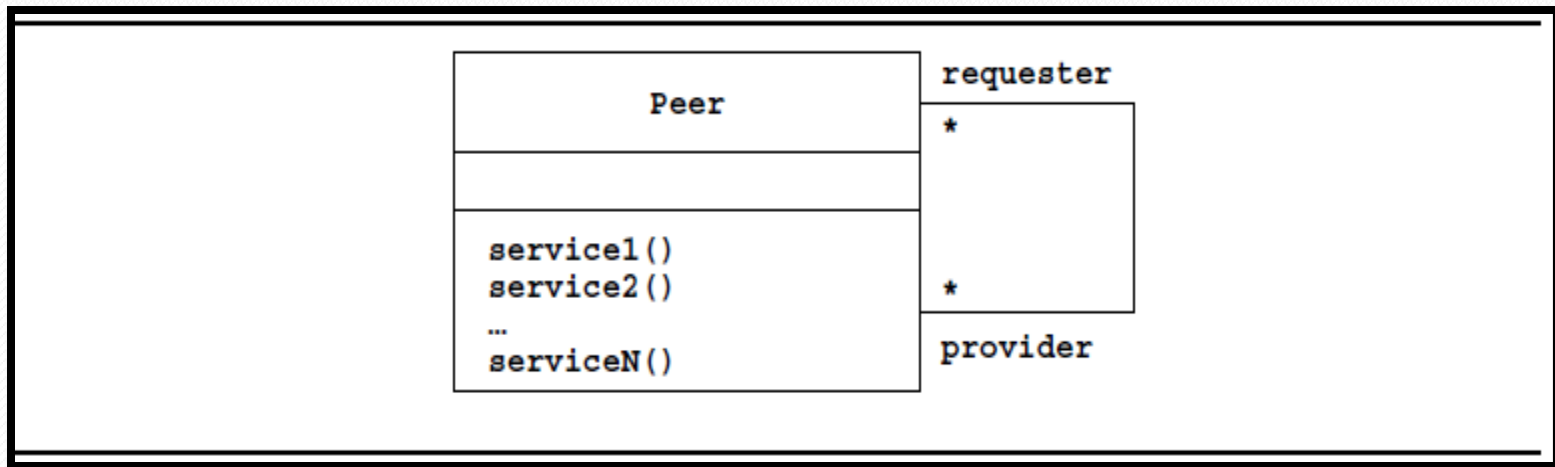


- Fig. Client/server architecture (UML class diagram).

Cont...

4. A peer-to-peer architecture

- It is a generalization of the client/server architecture in which subsystems can act both as client or as servers, in the sense that each subsystem can request and provide services.
- The control flow within each subsystem is independent from the others except for synchronizations on requests.

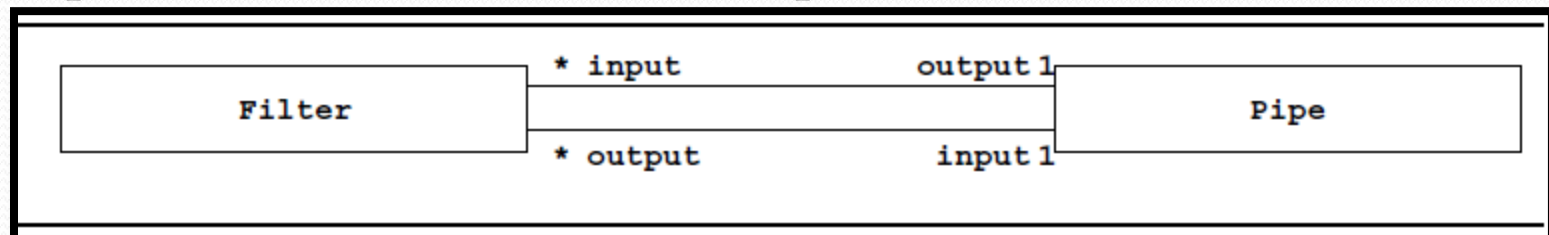


- Fig. Peer-to-peer architecture (UML class diagram).

Cont...

5. *Pipe and filter architecture*

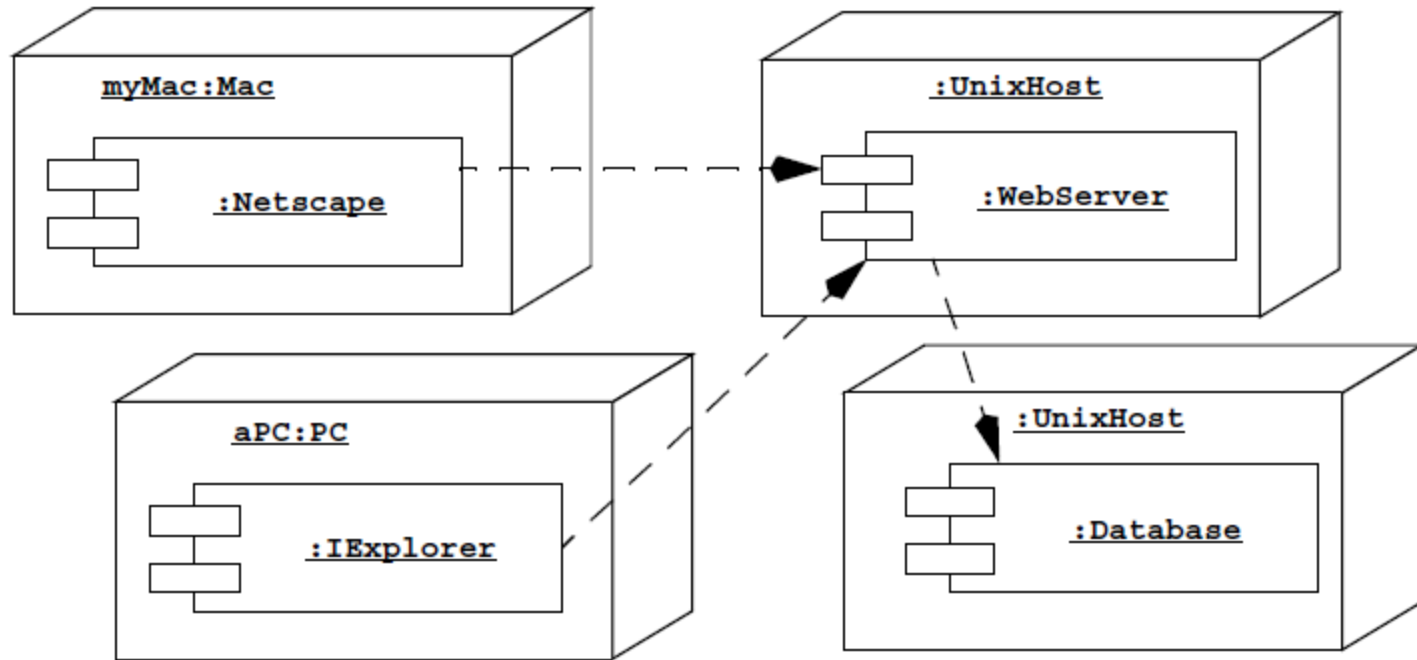
- In this architecture subsystems process data received from a set of inputs and send results to other subsystems via a set of outputs.
- The subsystems are called **filters**, and the associations between the subsystems are called **pipes**.
- Each filter only knows the content and the format of the data received on the input pipes, not the filters that produced them.
- Each filter is executed concurrently and synchronization is done via the pipes.
- Most operating systems and programming languages provide a data stream mechanism; .
- A Filter can have many inputs and outputs. A Pipe connects one of the outputs of a Filter to one of the inputs of another Filter.



UML deployment diagrams

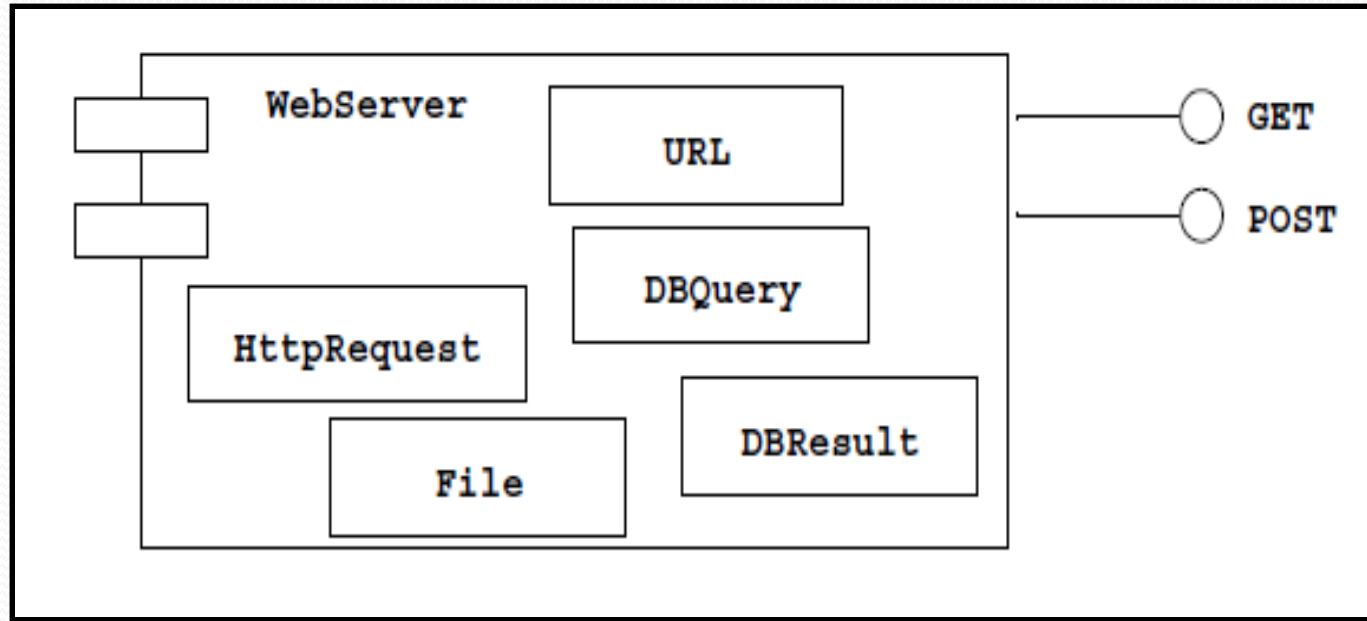
- **UML deployment diagrams:** are used to depict the relationship among **run-time components and hardware** nodes.
 - Components are **self-contained** entities that provide services to other components or actors. E. g. Web server, provides services to Web browsers.
 - In UML deployment diagrams, nodes are represented by **boxes** containing component icons.
 - Dependencies between components are represented by **dashed arrows**.

Cont...



- **Fig.** A UML deployment diagram representing the allocation of components to different nodes and the dependencies among components. Web browsers on PCs and Macs can access a WebServer that provides information from a Database.

Cont...



- **Fig.** Refined view of the WebServer component (UML deployment diagram). The WebServer component provides two interfaces to browsers: A browser can either request the content of a file referred by a URL (GET) or post the content of a form (POST). The WebServer component contains five classes: URL, HttpRequest, DBQuery, File, and DBResult.

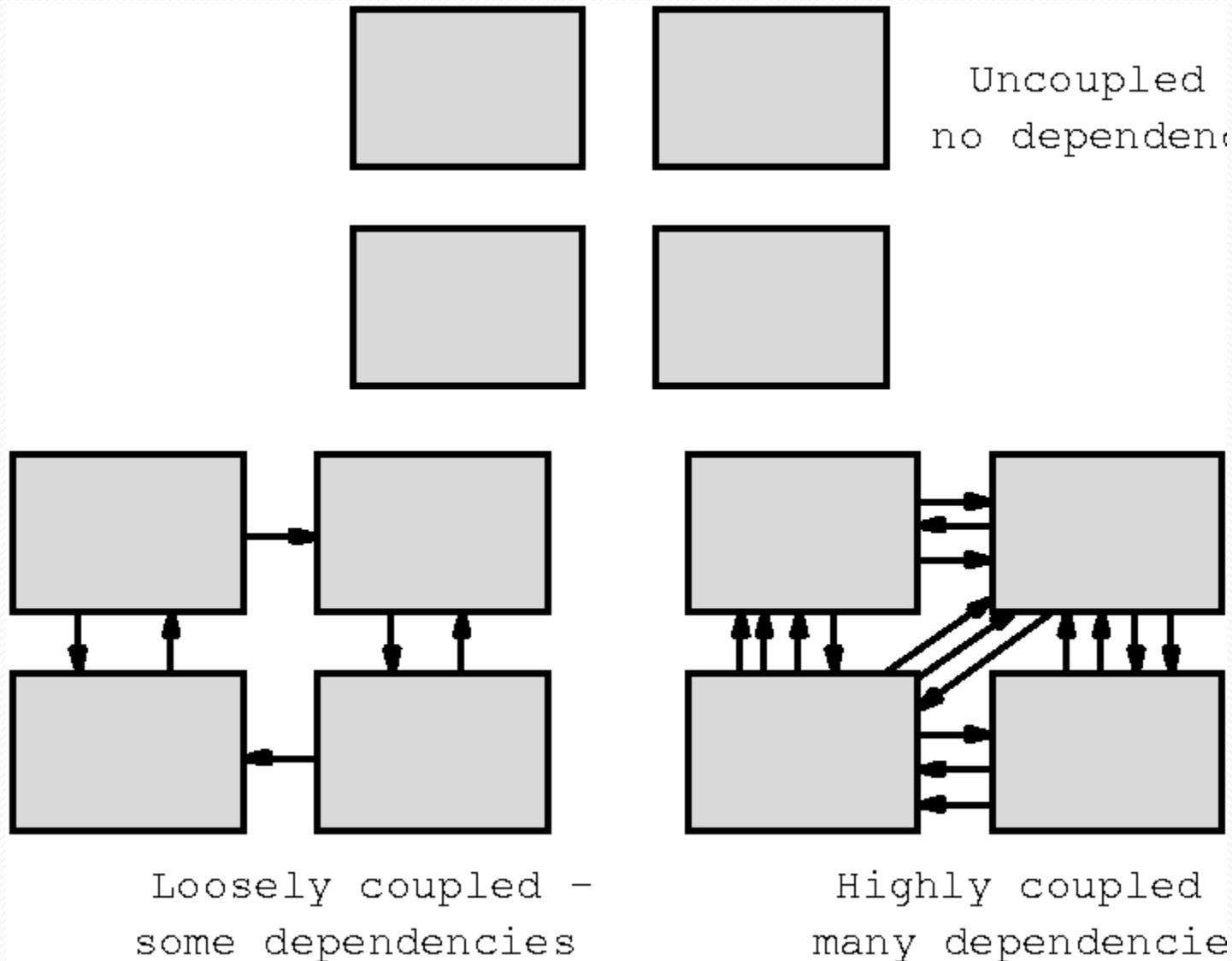
Structural decomposition

- Structural decomposition is concerned with developing a model of the design which shows the dynamic structure i.e. function calls
- This is not necessarily the same as the static composition structure
- The aim of the designer should be to derive (obtain) design units which are highly cohesive and loosely coupled
- In essence, a data flow diagram is converted to a structure chart

Decomposition guidelines

- The aim of the design process is to identify loosely coupled, highly cohesive functions. Each function should therefore do one thing and one thing only
 - Cohesion – the degree to which a module performs one and only one function.
 - Coupling – the degree to which a module is connected to other modules in the system.

Coupling and cohesion



Detailed design

- Concerned with producing a short design specification (minispec) of each function. This should describe the processing, inputs and outputs
- These descriptions should be managed in a data dictionary

Documenting system design

System Design Document

1. Introduction

1.1 Purpose of the system

1.2 Design goals

1.3 Definitions, acronyms, and abbreviations

1.4 References

1.5 Overview

2. Current software architecture

3. Proposed software architecture

3.1 Overview

3.2 Subsystem decomposition

3.3 Hardware/software mapping

3.4 Persistent data management

3.5 Access control and security

3.6 Global software control

3.7 Boundary conditions

4. Subsystem services

Glossary



Thank You!
Q?