Cornell University
Computing and Information Science

CS 5150 Software Engineering
12. System Architecture

William Y. Arms

Design

Design in Software Development

For a given set of requirements, the software development team must **design** a system that will meet those requirements.

This has many aspects:

- system architecture
- program design
- security
- performance

Design was also an important part of the lectures on usability.

In practice requirements and design are interrelated. In particular, working on the design often clarifies the requirements. This feedback is a strength of the iterative and agile methods of software development.

Creativity and Design

Creativity and design

System and program design are a particularly creative part of software development, as are user interfaces.

Above all strive for **simplicity**. The aim is find simple ways to implement complex requirements. You hope that people will describe your designs as “elegant”, “easy to implement, test, and maintain.”

Software development as a craft

Software development is a **craft**. Software developers have a variety of **tools** that can be used in design, but you have to select the appropriate tool for a given implementation.

System Architecture

System architecture is the overall design of a system:

- Computers and networks (e.g., in-house computers, cloud services)
- Interfaces and protocols (e.g., http, IMAP, ODBC)
- Databases (e.g., relational, distributed)
- Security (e.g., Shibboleth)
- Operations (e.g., backup, archiving, audit trails)

At this stage of the development process, you should also be selecting:

- Software environments (e.g., languages, database systems, class frameworks)
- Testing frameworks

Models for System Architecture

Our models for systems architecture are based on UML.

For every system, there is a choice of models. Choose the models that best model the system and are clearest to everybody.

In UML, every model must have both a diagram and a supporting specification.

The lectures provide diagrams that give an outline of the system, without the supporting specifications.

The diagrams shows the relationships among parts of the system, but much, much more detail is needed to specify a system.

Subsystems

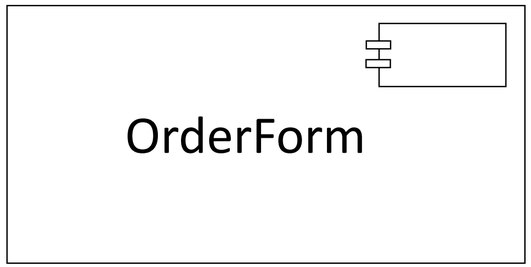
Subsystem

A subsystem is a grouping of elements that form part of a system.

- **Coupling** is a measure of the dependencies between two subsystems. If two systems are strongly coupled, it is hard to modify one without modifying the other.
- **Cohesion** is a measure of dependencies within a subsystem. If a subsystem contains many closely related functions its cohesion is high.

An ideal division of a complex system into subsystems has low coupling between subsystems and high cohesion within subsystems.

Component



A **component** is a replaceable part of a system that conforms to and provides the realization of a set of interfaces.

A component can be thought of as an implementation of a subsystem.

UML definition of a component

"A distributable piece of implementation of a system, including software code (source, binary, or executable), but also including business documents, etc., in a human system."

Components as Replaceable Elements

Components allow systems to be assembled from **binary replaceable elements.**

- A component is bits not concepts.
- A component can be replaced by any other component(s) that conforms to the interfaces.
- A component is part of a system.
- A component provides the realization of a set of interfaces.

Components and Classes

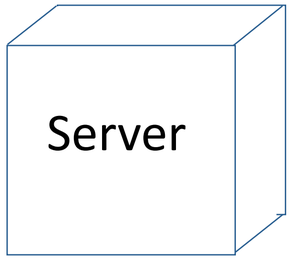
Classes represent logical abstractions. They have attributes (data) and operations (methods). Classes can be combined to form programs.

Components have operations that are reachable only through interfaces. Components can be combined to form systems.

Package

A **package** is a general-purpose mechanism for organizing elements into groups.

Node



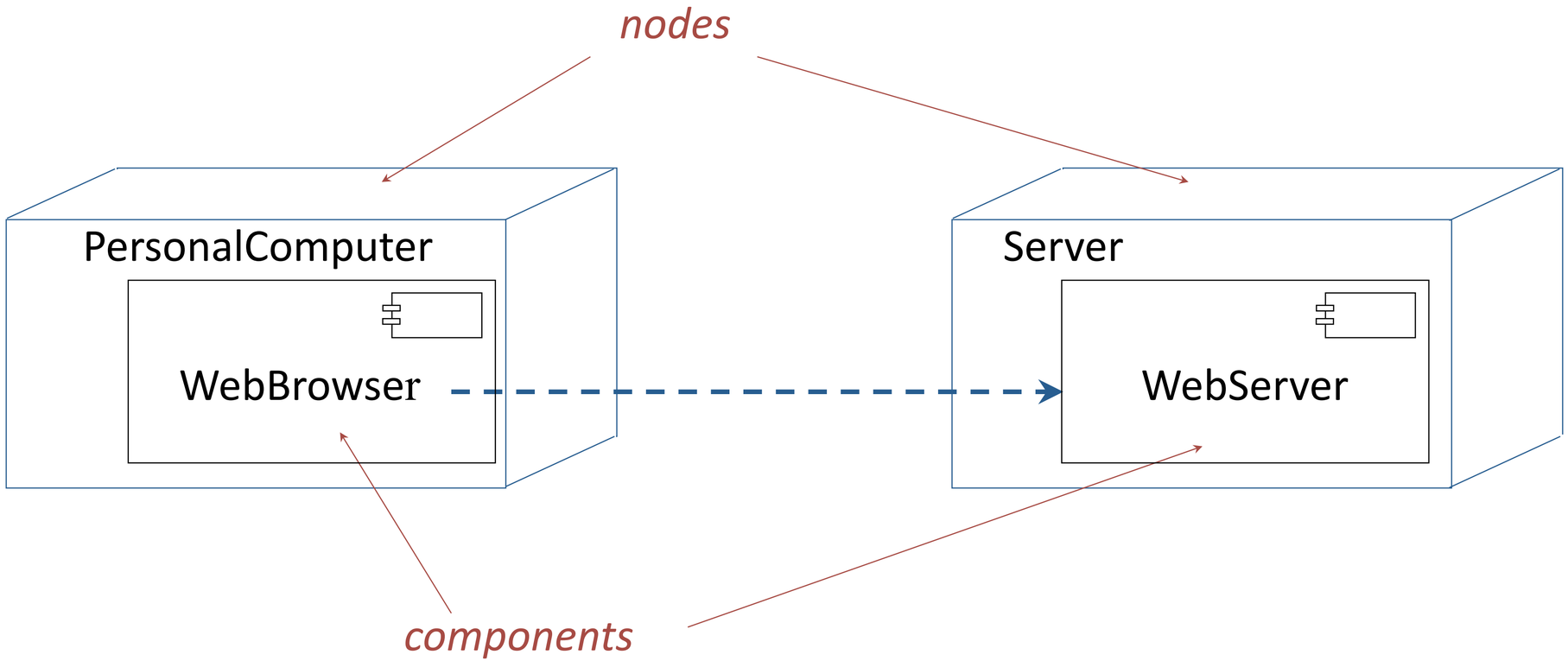
A **node** is a physical element that exists at run time and provides a computational resource, e.g., a computer, a smartphone, a router.

Components may live on **nodes**.

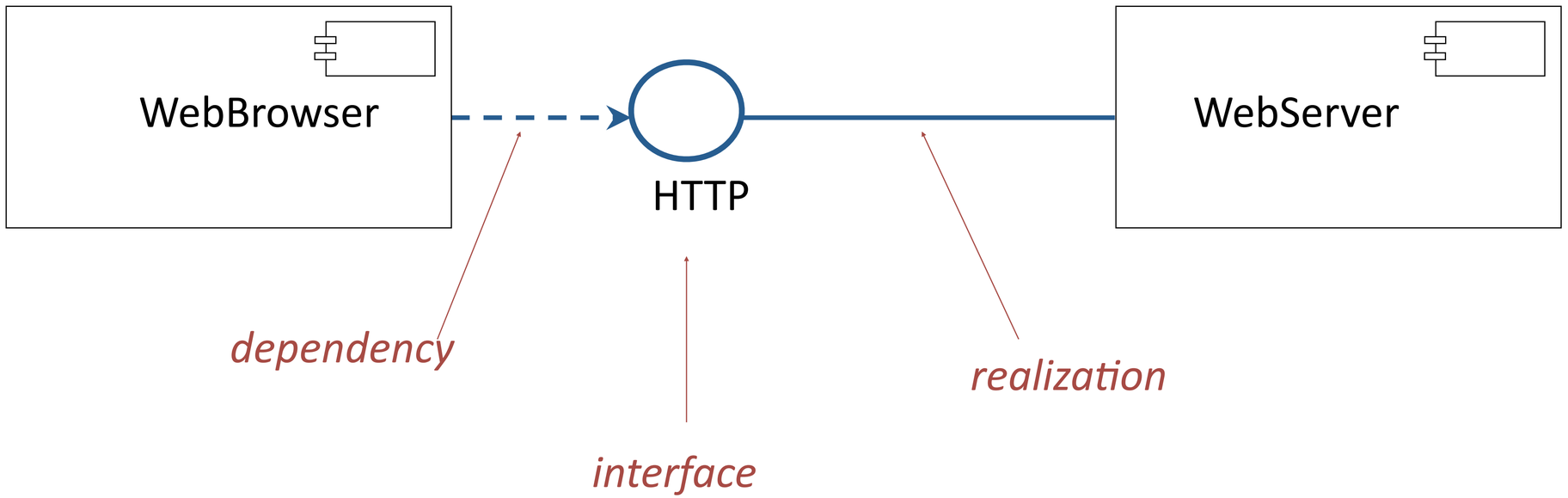
Example: Simple Web System

- Static pages from server
- All interaction requires communication with server

Deployment Diagram

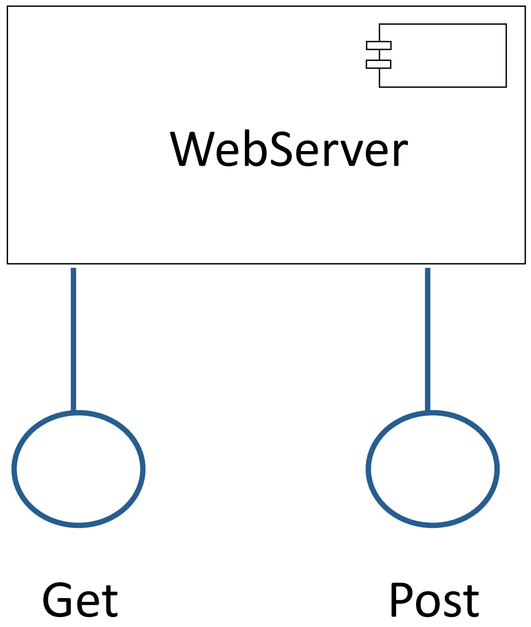


Component Diagram: Interfaces



Application Programming Interface (API)

An **API** is an interface that is realized by one or more components.



Architectural Styles

An **architectural style** is system architecture that recurs in many different applications.

See:

- Mary Shaw and David Garlan, *Software architecture: perspectives on an emerging discipline*. Prentice Hall, 1996
- David Garlan and Mary Shaw, *An Introduction to Software Architecture*. Carnegie Mellon University, 1994

  http://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf

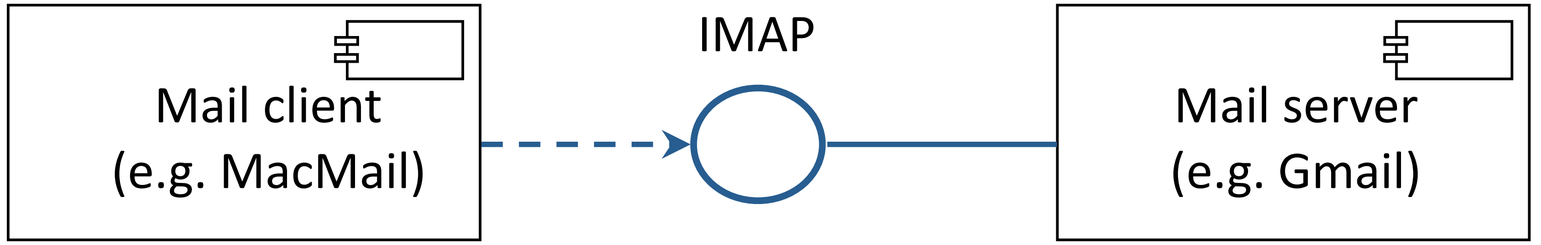
Architectural Style: Pipe

Example: A three-pass compiler

Output from one subsystem is the input to the next.

Architectural Style: Client/Server

Example: A mail system

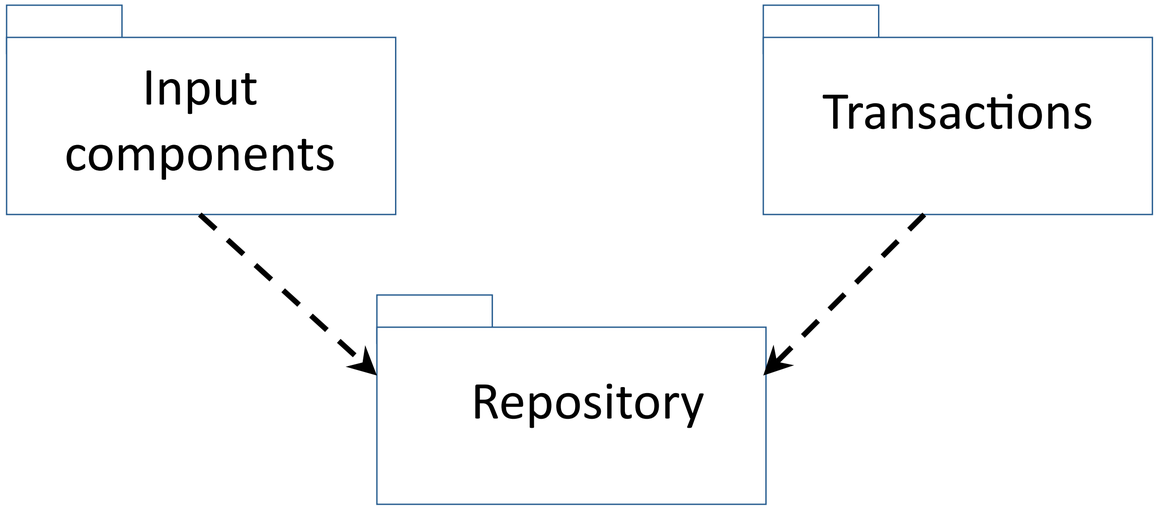


The control flows in the client and the server are independent. Communication between client and server follows a protocol.

Because components are binary replaceable elements, either the client or the server can be replaced by another component that implements the protocol.

In a peer-to-peer architecture, the same components act as both client and server.

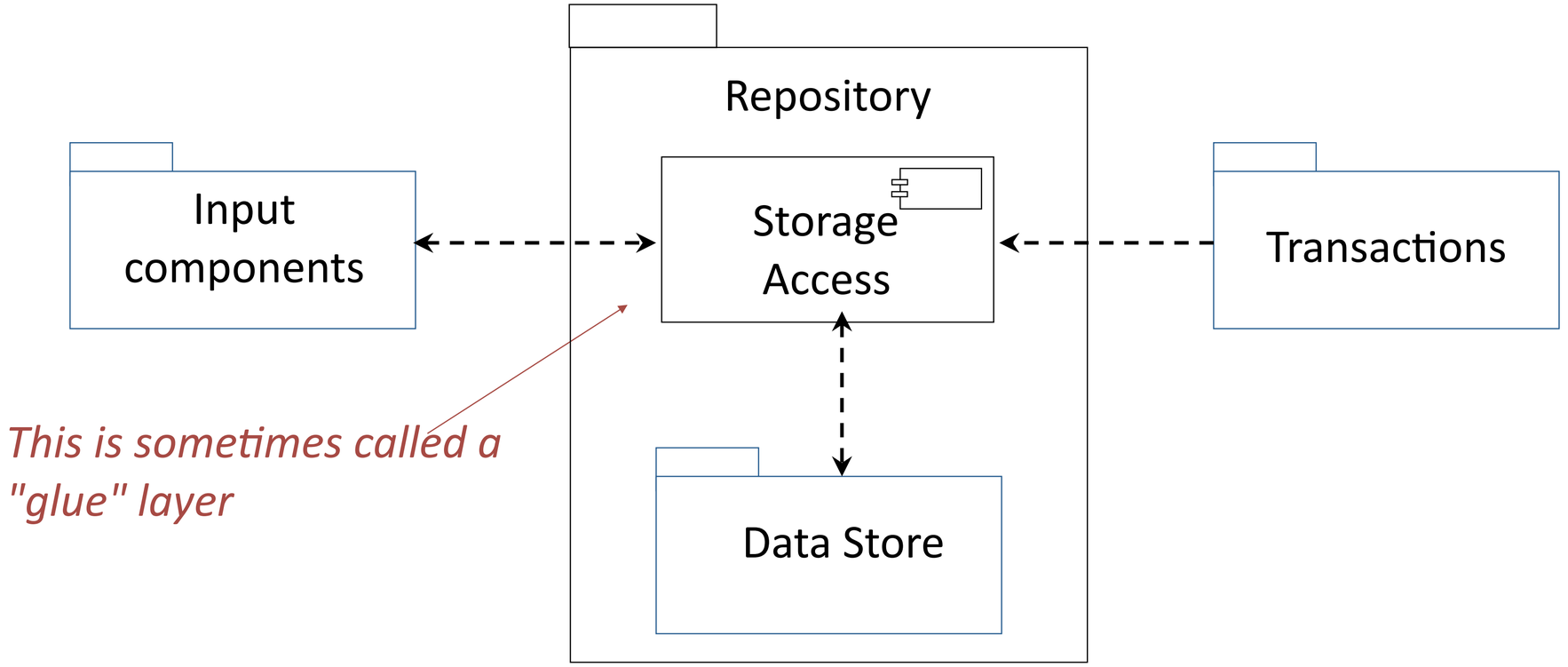
Architectural Style: Repository



Advantages: Flexible architecture for data-intensive systems.

Disadvantages: Difficult to modify repository since all other components are coupled to it.

Architectural Style: Repository with Storage Access Layer



Advantages: Data Store subsystem can be changed without modifying any component except the Storage Access.

Time-Critical Systems

A **time-critical** (real time) system is a software system whose correct functioning depends upon the results produced and the time at which they are produced.

- A hard real time system fails if the results are not produced within required time constraints,

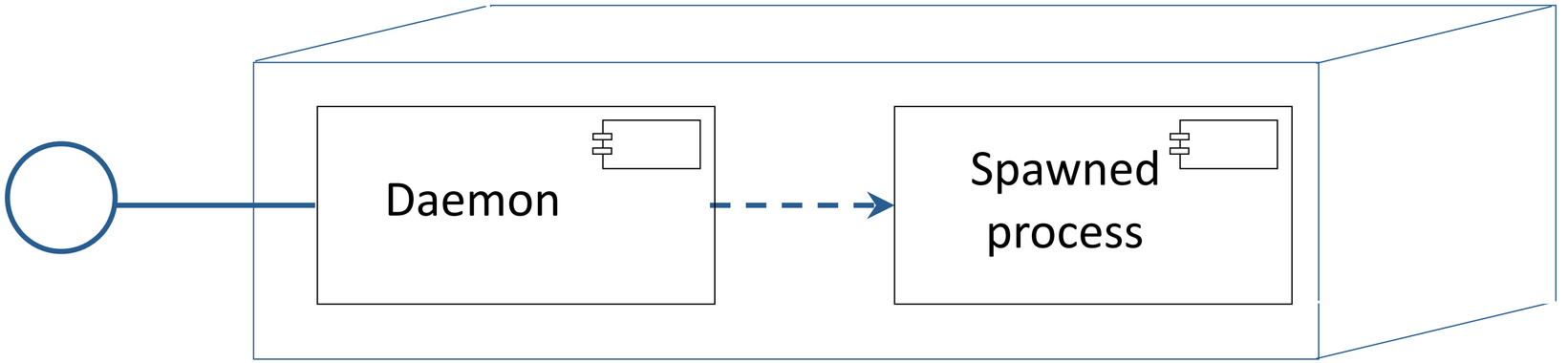
  e.g., a fly-by-wire control system for an airplane must respond within specified time limits.

- A soft real time system is degraded if the results are not produced within required time constraints,

  e.g., a network router is permitted to time out or lose a packet.

Time Critical System: Architectural Style - Daemon

A **daemon** is used when messages might arrive at closer intervals than the the time to process them.



Example: Web server

The daemon listens at port 80

When a message arrives it:

- spawns a processes to handle the message
- returns to listening at port 80

Architectural Styles for Distributed Data

Replication:

Several copies of the data are held in different locations.

Mirror: Complete data set is replicated

Cache: Dynamic set of data is replicated (e.g., most recently used)

With replicated data, the biggest problems are concurrency and consistency.

Example: The Domain Name System

For details of the protocol read:

Paul Mockapetris, "Domain Names - Implementation and Specification". IETF Network Working Group, *Request for Comments*: 1035, November 1987.

http://www.ietf.org/rfc/rfc1035.txt?number=1035

An Old Exam Question

A company that makes sports equipment decides to create a system for selling sports equipment online. The company already has a ***product database*** *with description, marketing information, and prices of the equipment that it manufactures.*

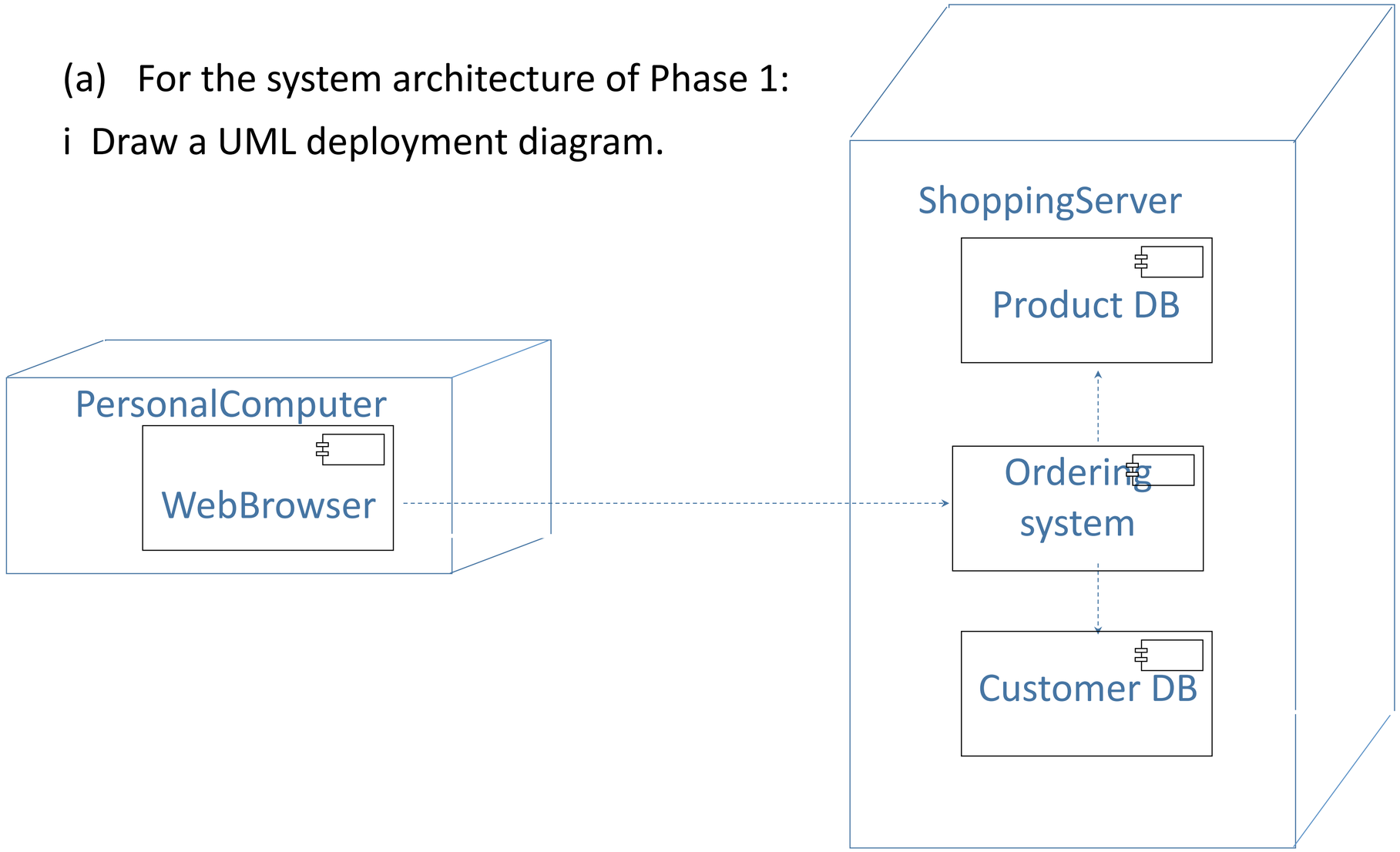
To sell equipment online the company will need to create: a ***customer database****, and an* ***ordering system*** *for online customers.*

The plan is to develop the system in two phases. During Phase 1, simple versions of the customer database and ordering system will be brought into production. In Phase 2, major enhancements will be made to these components.

An Old Exam Question

(a) For the system architecture of Phase 1:

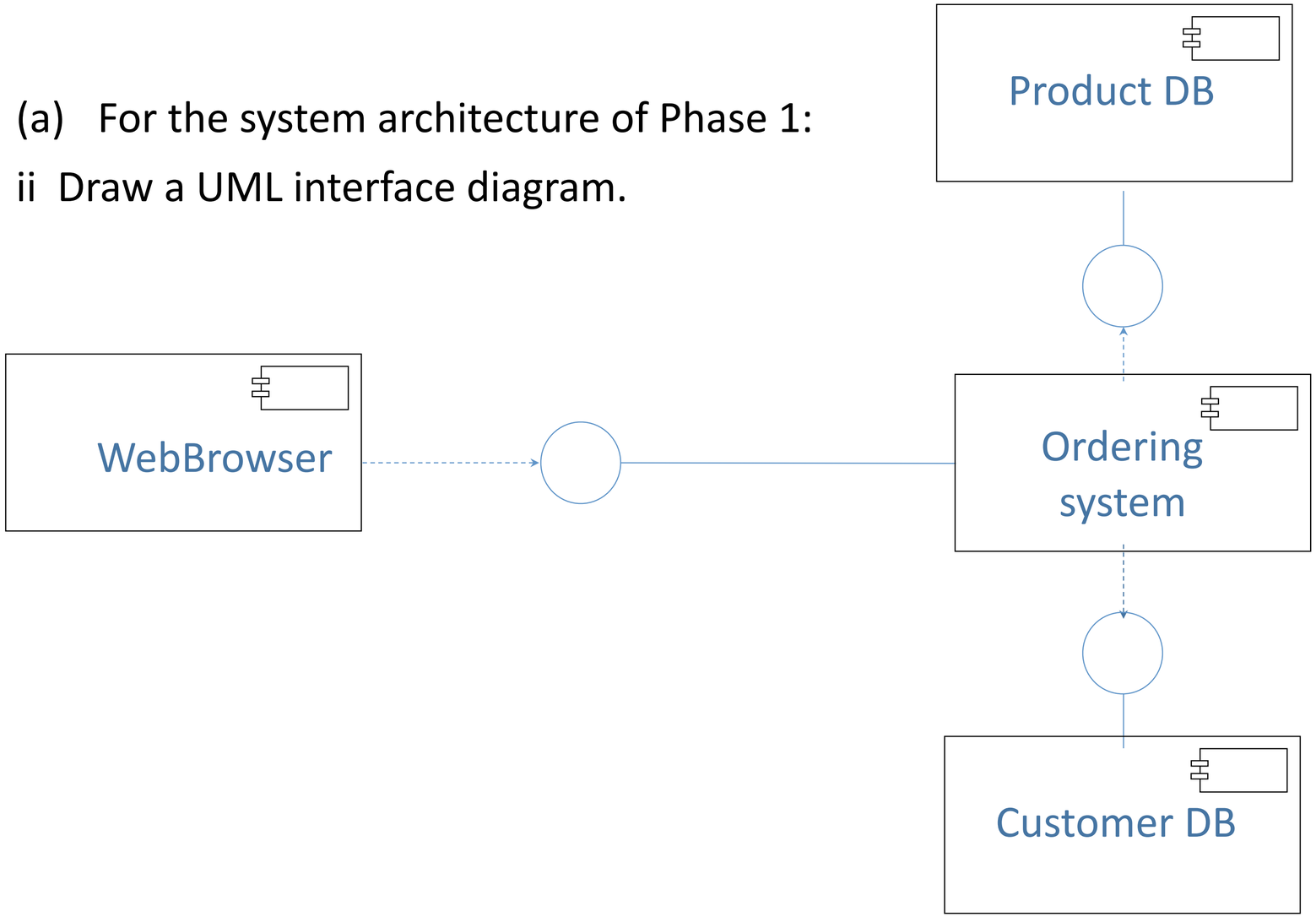
i Draw a UML deployment diagram.



An Old Exam Question

(a) For the system architecture of Phase 1:

ii Draw a UML interface diagram.



An Old Exam Question

(b) For Phase 1:

i What architectural style would you use for the customer database?

Repository with Storage Access Layer

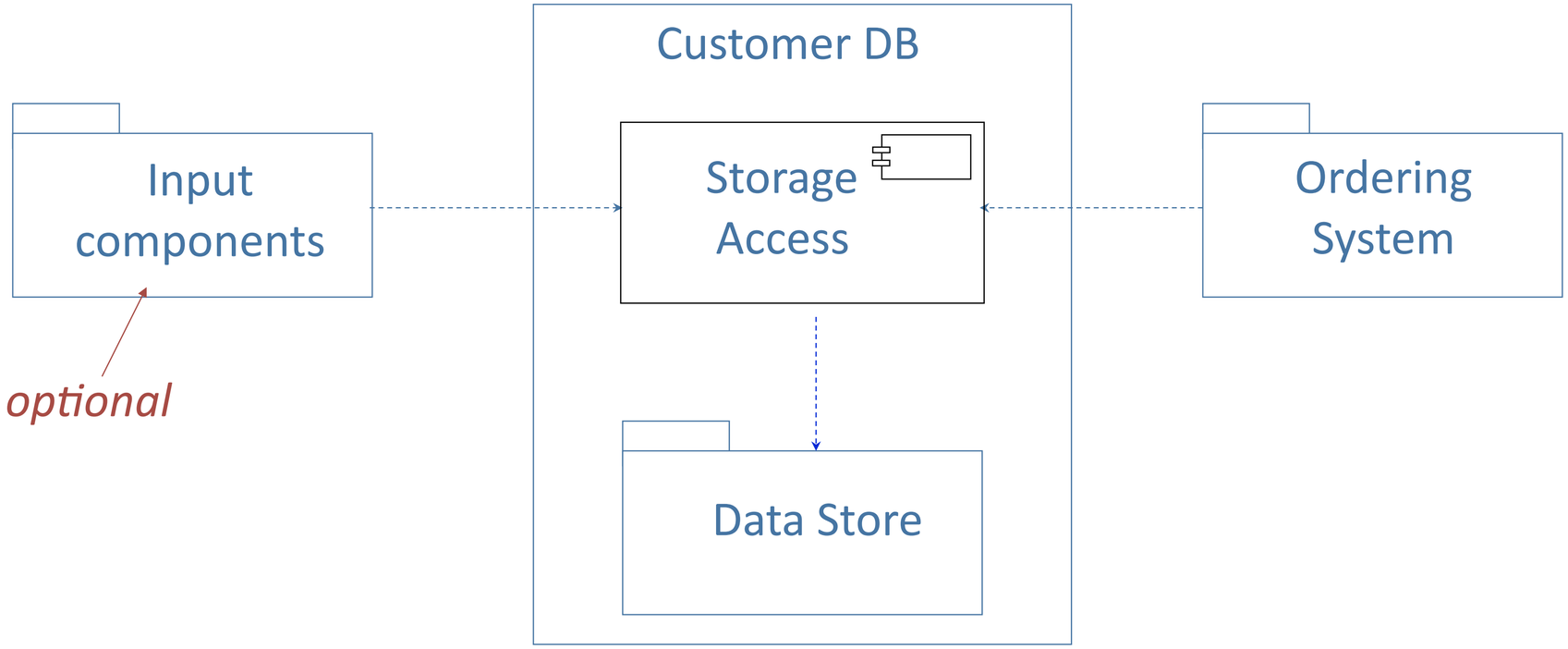
ii Why would you choose this style?

It allows the database to be replaced without changing the applications that use the database.

An Old Exam Question

(b) For Phase 1:

iii Draw an UML diagram for this architectural style showing its use in this application.



Cornell University
Computing and Information Science

CS 5150 Software Engineering
12. System Architecture

End of Lecture