FARABI UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Lecture 10.
UML Component Diagrams

Software modules, contracts and connections

Dr. Bazargul Matkerim, Professor-Researcher

Lecture goals and plan

Learning outcomes

Identify components and responsibilities.

Read provided and required interfaces.

Distinguish dependencies, assembly and delegation.

Relate components to artifacts and nodes.

Lecture plan

1. Components and contracts
2. Interfaces, dependencies and ports
3. Assembly and delegation
4. Course-feedback case study
5. Deployment and review

Components as modular units

Component meaning

A component encapsulates part of a system.

It exposes services through interfaces.

It can depend on services supplied by other components.

Architectural purpose

Separate responsibilities.

Hide implementation decisions.

Support reuse and compatible replacement.

Show structural dependencies.

Component notation



Keyword notation



Component icon notation

Use a meaningful component name and a consistent notation.

Case study: course-feedback system

User requirements

Students read course information.

They create, update and save feedback for courses in their schedule.

Professors view aggregated statistics.

Deployment context

Student applications run on client PCs.

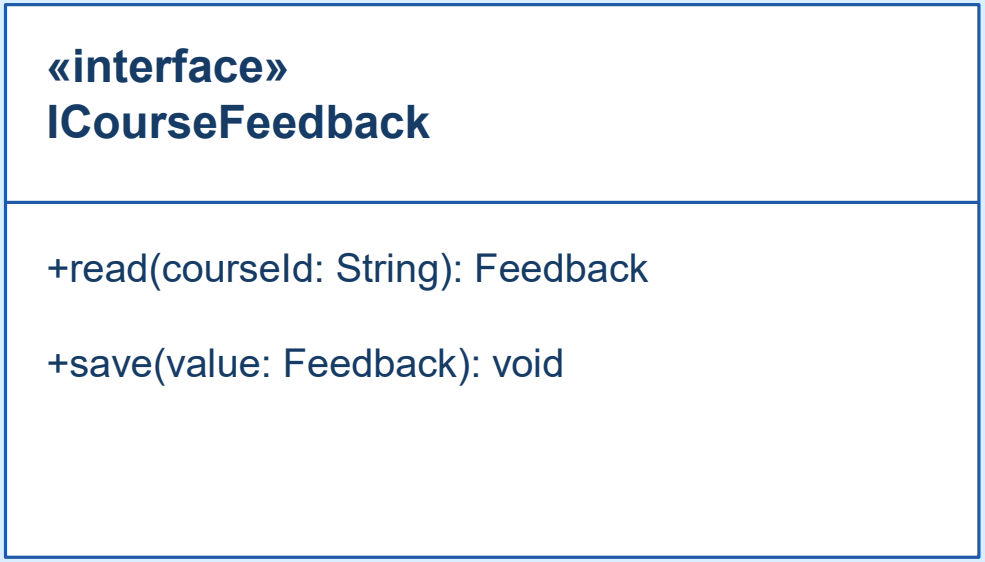
A manager application runs in the office.

Servers host course services and persistent data.

Candidate component responsibilities

Component	Responsibility
Student Client	Course selection and feedback interaction
Manager Client	Access to statistical reports and office functions
Course Service	Course feedback rules and reporting services
Data Access	Persistence operations behind a data contract
Security Service	Authentication and authorization services
Office Service	Integration with office functions

Interfaces define service contracts



A contract states what clients can request and what providers promise.

Define operation signatures, meaning, constraints and error behavior.

Keep implementation details inside the component.

Provided and required interfaces



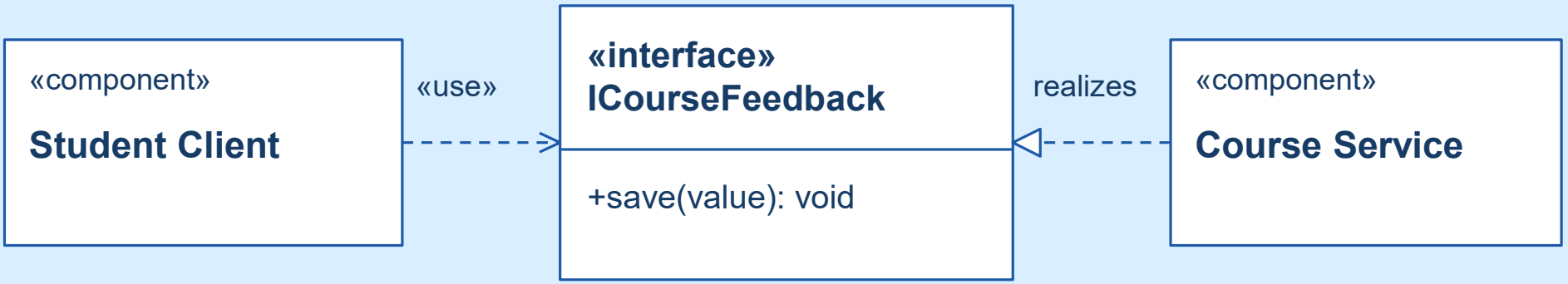
Provided interface

The component offers this service.

Required interface

The component needs this service.

Expanded interface relationships



Usage points to the required contract. Realization points to the implemented contract.

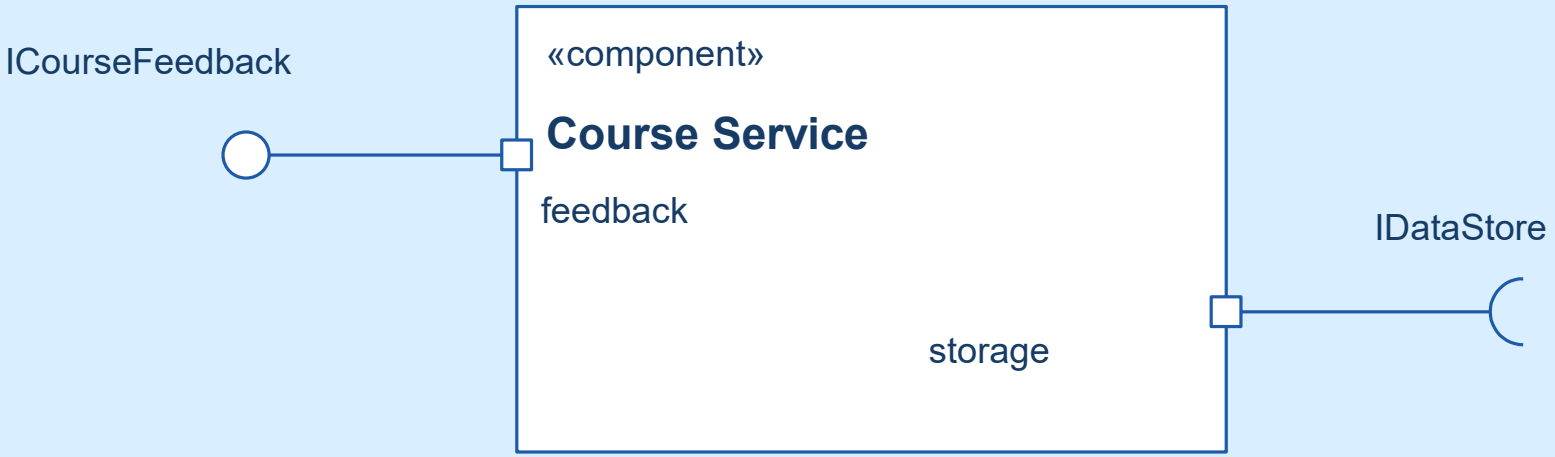
Usage dependencies



Client: depends on the supplier

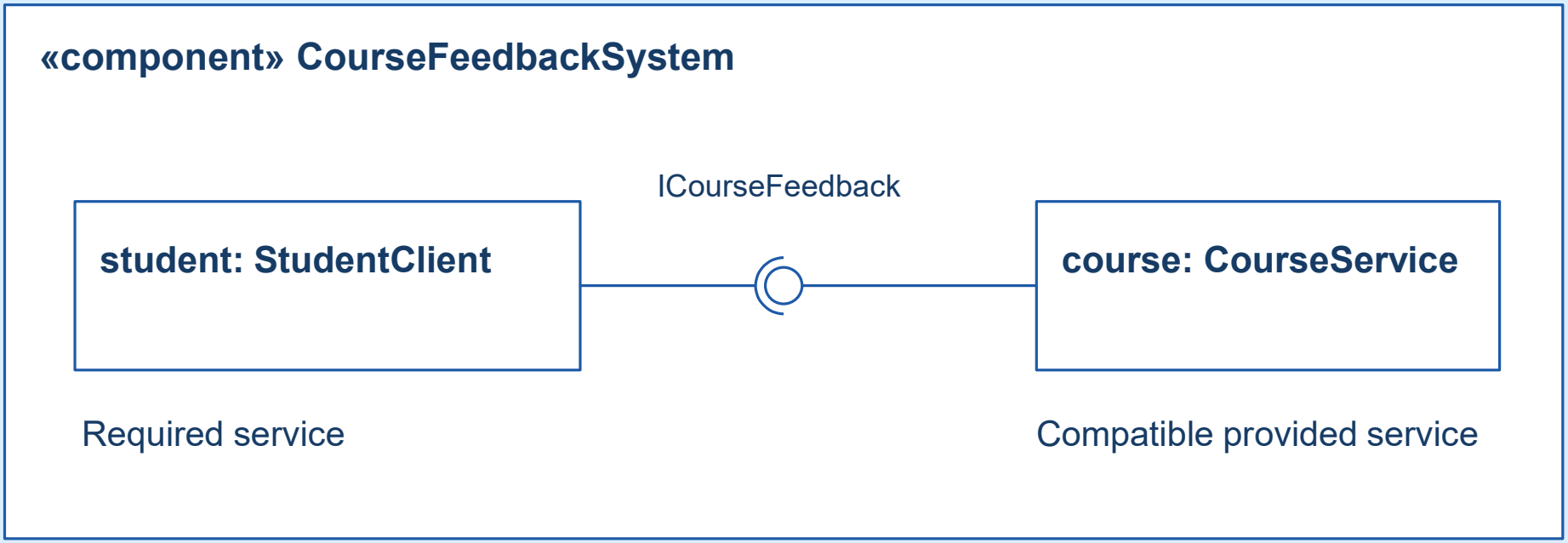
Supplier: provides a needed capability

Ports as interaction points

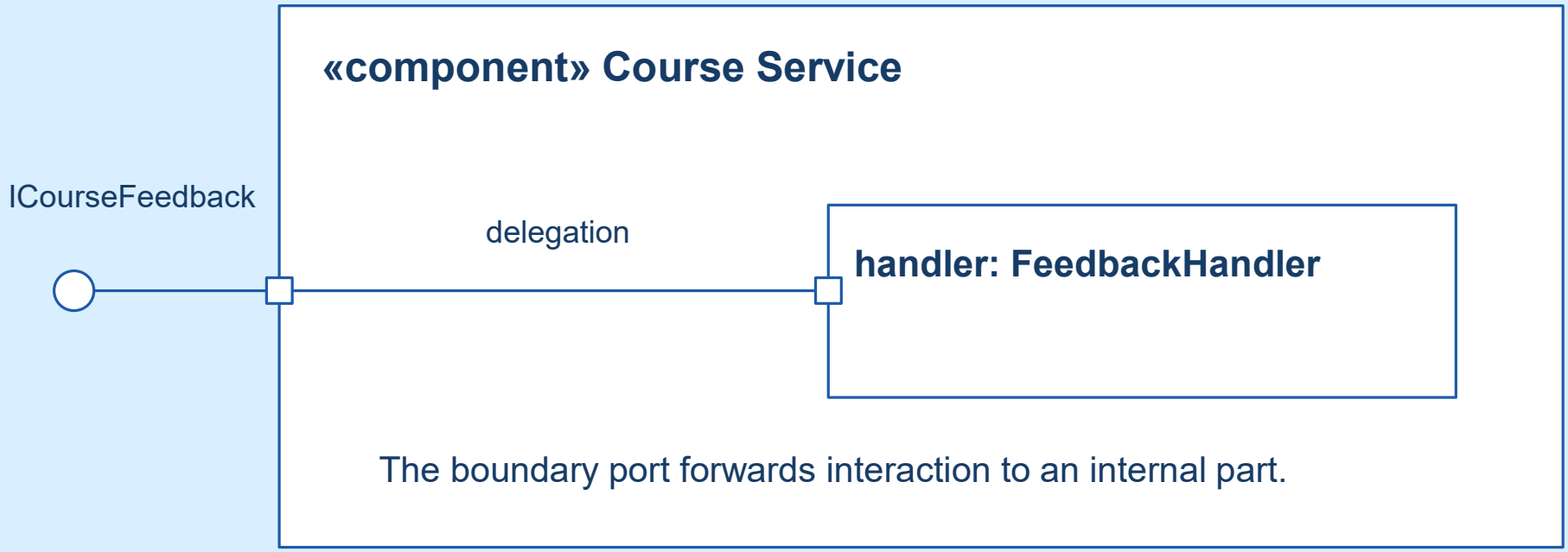


A named port can expose provided and required interfaces.

Assembly connectors



Delegation connectors



External and internal component views

External view

Shows the component boundary and its public contracts.

Clients see what the component provides and requires.

Implementation details remain hidden.

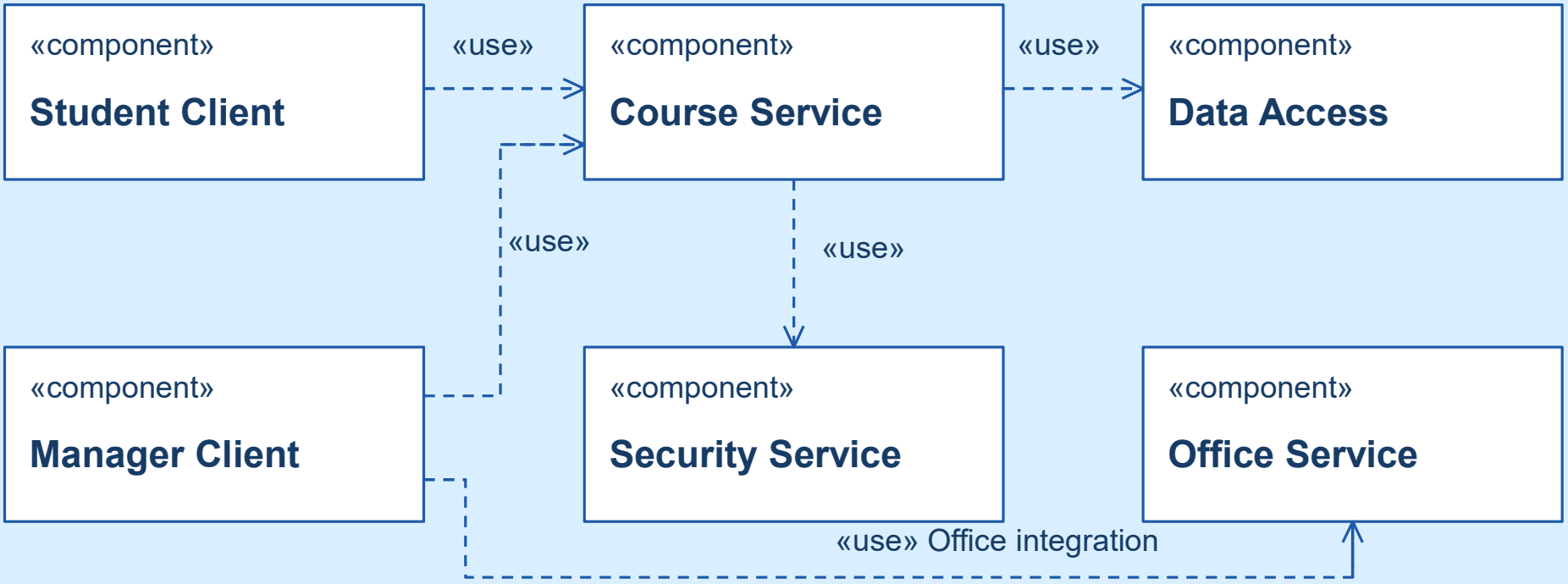
Internal view

Shows parts, realizing elements and their connections.

Delegation maps boundary services to internal responsibilities.

Components may contain other components.

Case study: component dependencies



Compatible component replacement

What must remain compatible

Provided operation signatures and data types.

Promised behavior and error handling.

Authorization and quality constraints.

Services required from the environment.

Case-study change

Replace the Data Access implementation.

Keep the IDataStore contract stable.

Verify behavior with contract tests.

Review migrations and performance assumptions.

Components, artifacts and manifestation



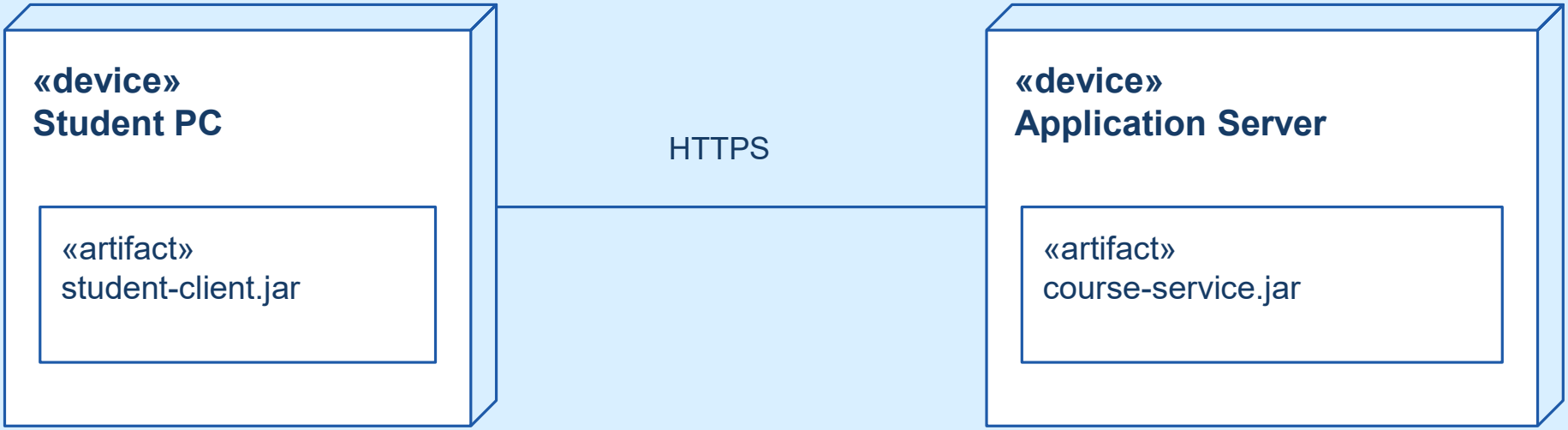
Component

A modular design element with a service contract.

Artifact

A physical information item, such as a deployable file.

Deployment view of the case study



Illustrative deployment slice. The manager client and database node are omitted.

Review and classroom practice

Model review

Does each component have one clear responsibility?

Are interface directions correct?

Are assembly and delegation distinct?

Are artifacts separate from components?

Practice: reporting service

Add a Statistics Service component.

Define the report interface it provides.

Connect the manager’s required interface.

Identify its data dependency and deployment artifact.

Reading and references

Core reading

Dennis, A., Wixom, B. H., & Tegarden, D. (2021).
Systems Analysis and Design: An Object-Oriented Approach
with UML, 6th ed. Wiley. ISBN 9781119559917.

UML notation reference

Object Management Group (2017). Unified Modeling Language,
Version 2.5.1. Interfaces, components and deployments
(Chapters 10–11 and 19).

Case study adapted from Veronica Carrega, Component Diagram in UML 2.0 (supplied slides).