Software Engineering Essentials

System Design

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics

Learning goals

1) Remember the eight issues of the system design activities

2) Understand the transformation from analysis to design

3) Understand the issues design goals, subsystem decomposition and hardware/software mapping

Motivation - Why is design so difficult?

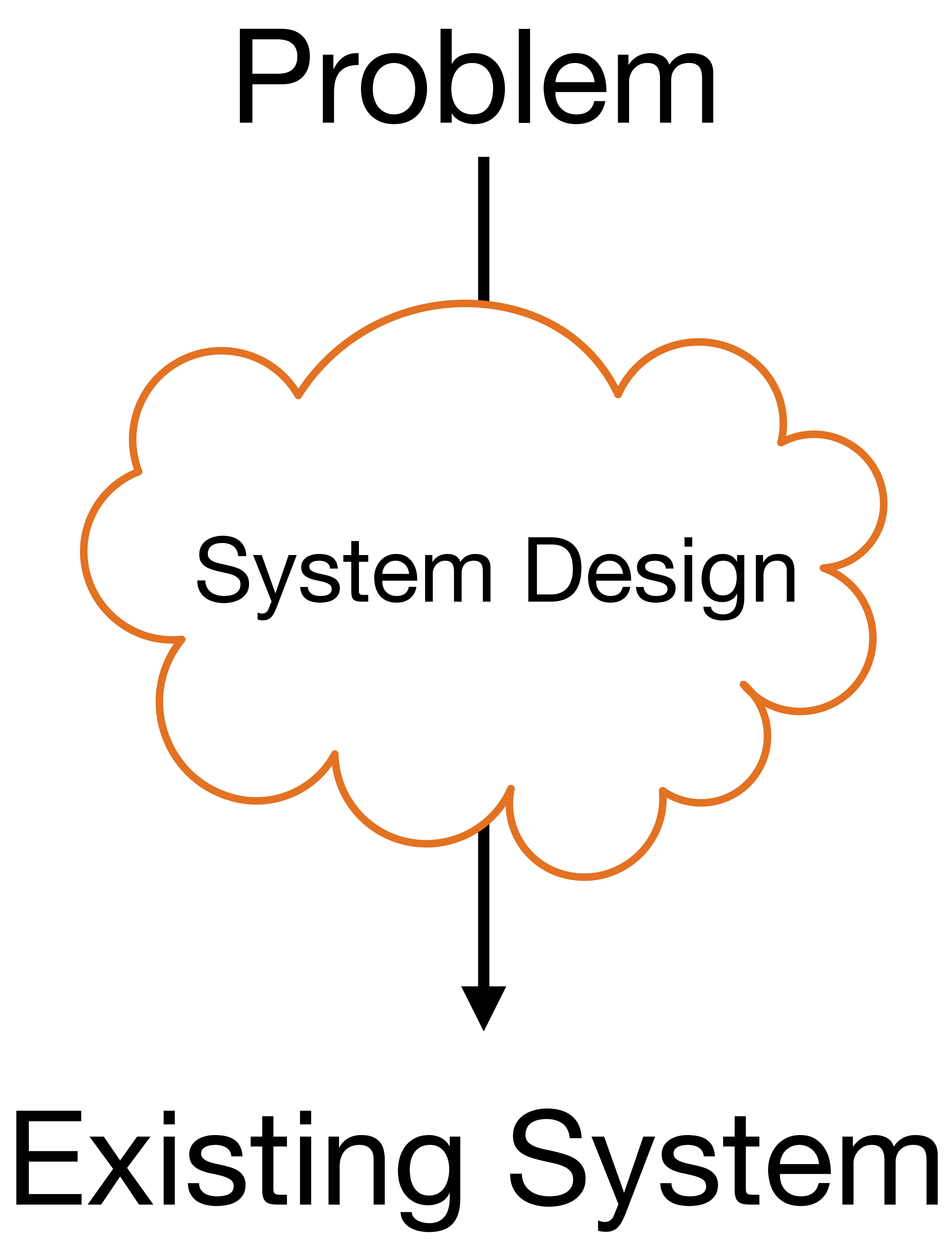
- Analysis: focuses on the application domain
- Design: focuses on the solutions domain
  - The solution domain is changing very rapidly
    - Halftime knowledge in software engineering: 3-5 years
    - Cost of hardware rapidly sinking

also called half-life of knowledge

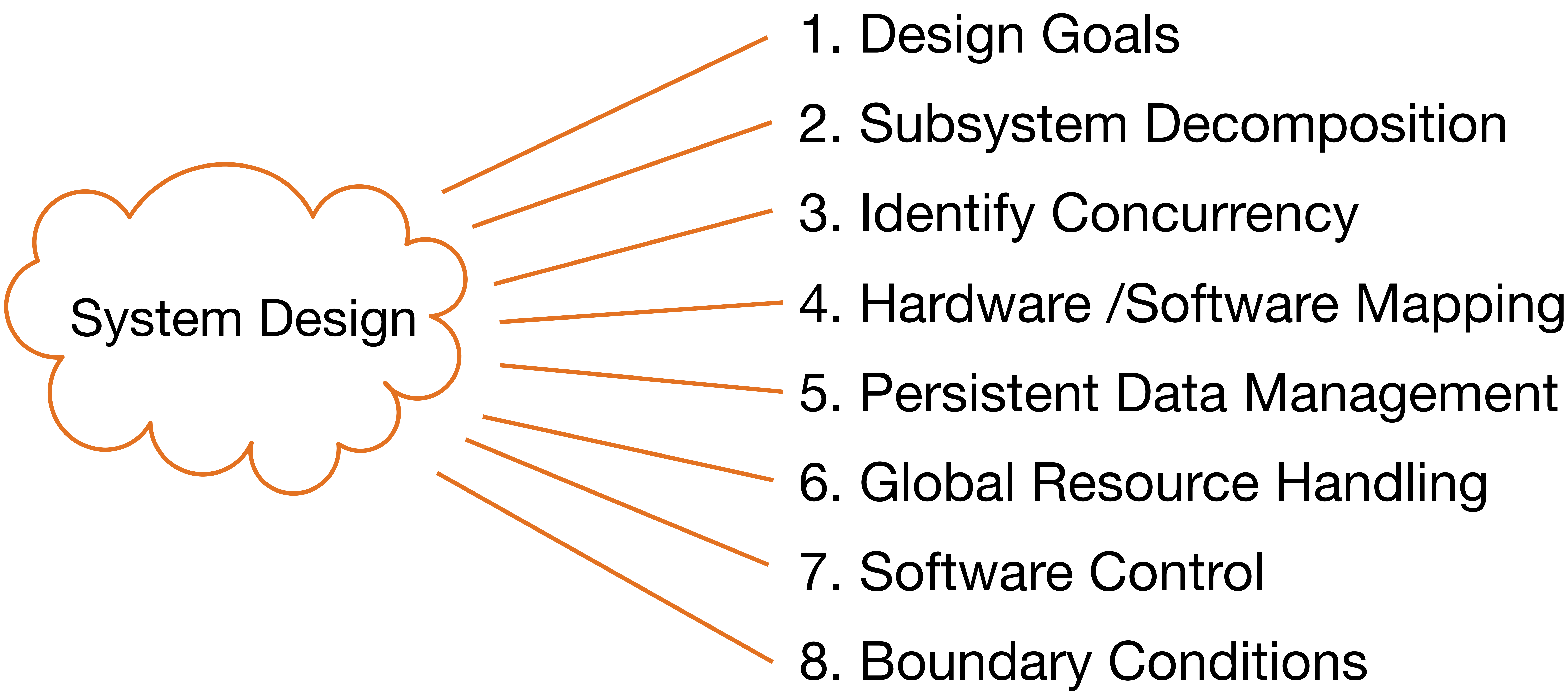
The scope of system design

- Bridge the gap between a problem and an existing system in a manageable way

- How? Divide & Conquer
  - Identify design goals
  - Model the new system design as a set of subsystems
  - Address the major design goals



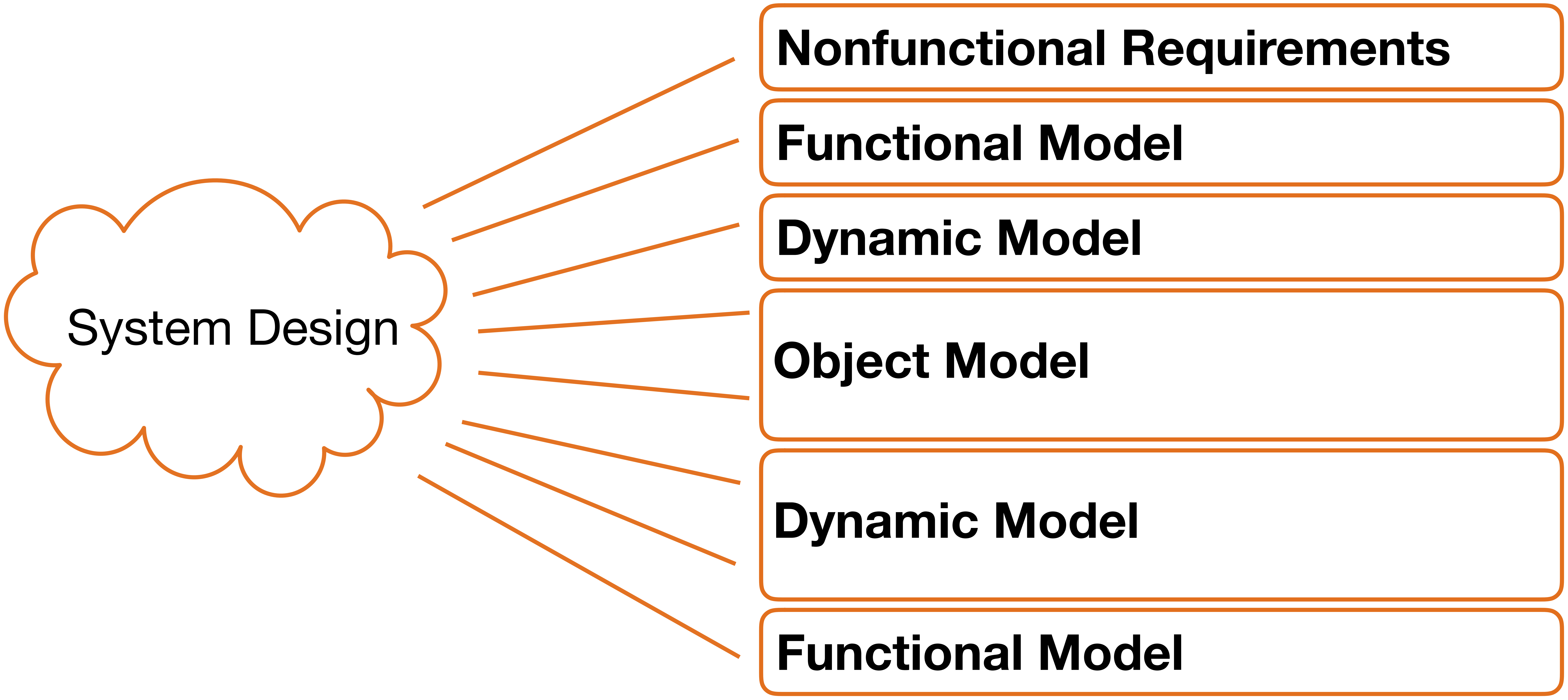
System Design: eight Issues



System Design

1. Design Goals
2. Subsystem Decomposition
3. Identify Concurrency
4. Hardware /Software Mapping
5. Persistent Data Management
6. Global Resource Handling
7. Software Control
8. Boundary Conditions

From analysis to design



System Design

Nonfunctional Requirements

Functional Model

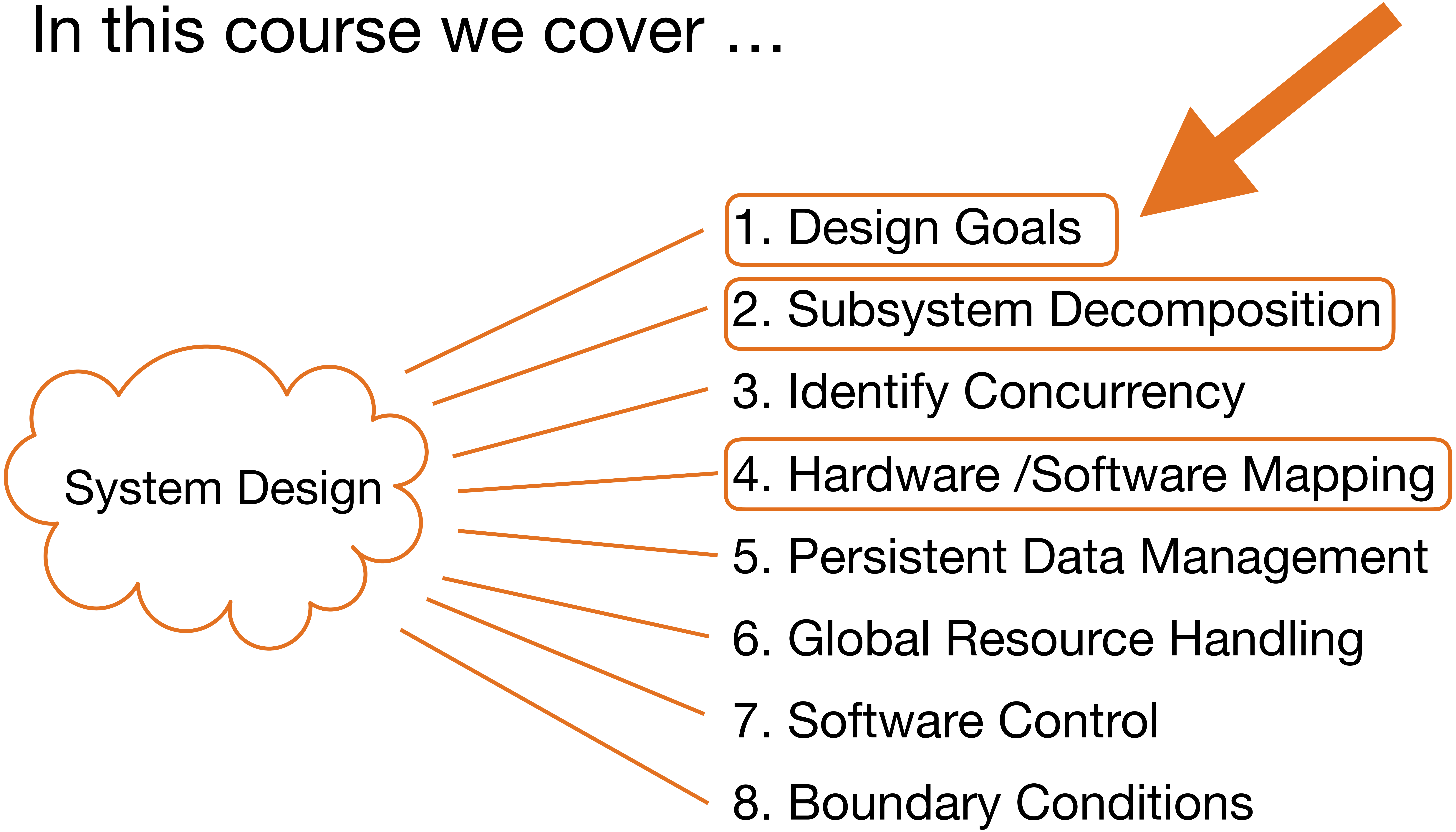
Dynamic Model

Object Model

Dynamic Model

Functional Model

In this course we cover …



System Design

1. Design Goals
2. Subsystem Decomposition
3. Identify Concurrency
4. Hardware /Software Mapping
5. Persistent Data Management
6. Global Resource Handling
7. Software Control
8. Boundary Conditions

Design goals

- Design goals govern the system design activities
- Any nonfunctional requirement is a design goal
- Additional stakeholder goals are formulated with respect to
  - Design methodology
  - Design metrics
  - Implementation goals
- Design goals often conflict with each other

→ Typical design goal trade-offs

Typical design goal trade-offs

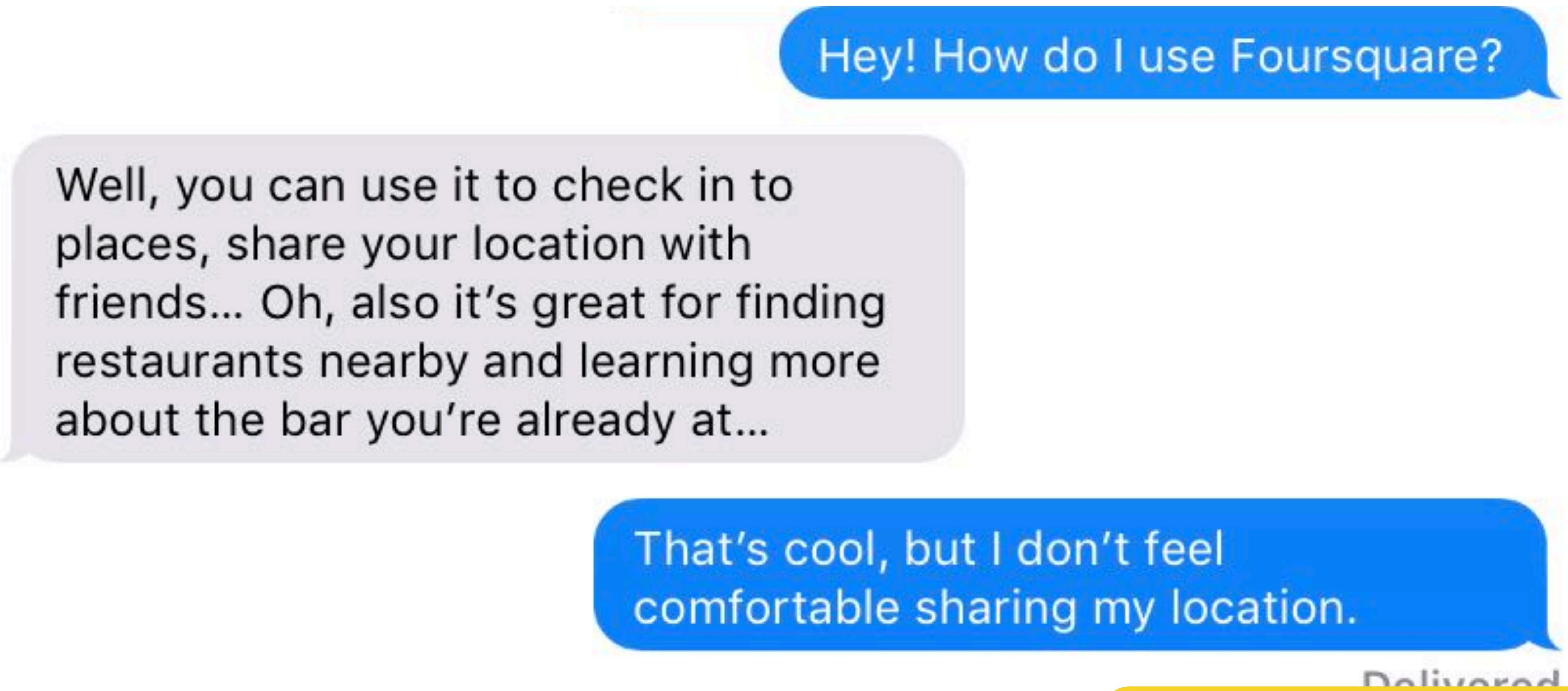
- Functionality vs. usability
- Cost vs. robustness
- Efficiency vs. portability
- Rapid development vs. functionality
- Cost vs. reusability
- Backward compatibility vs. readability

Example: functionality vs. usability

Is a system with 100 functions usable?

Some systems are not even usable with two functions.

Foursquare for searching venues

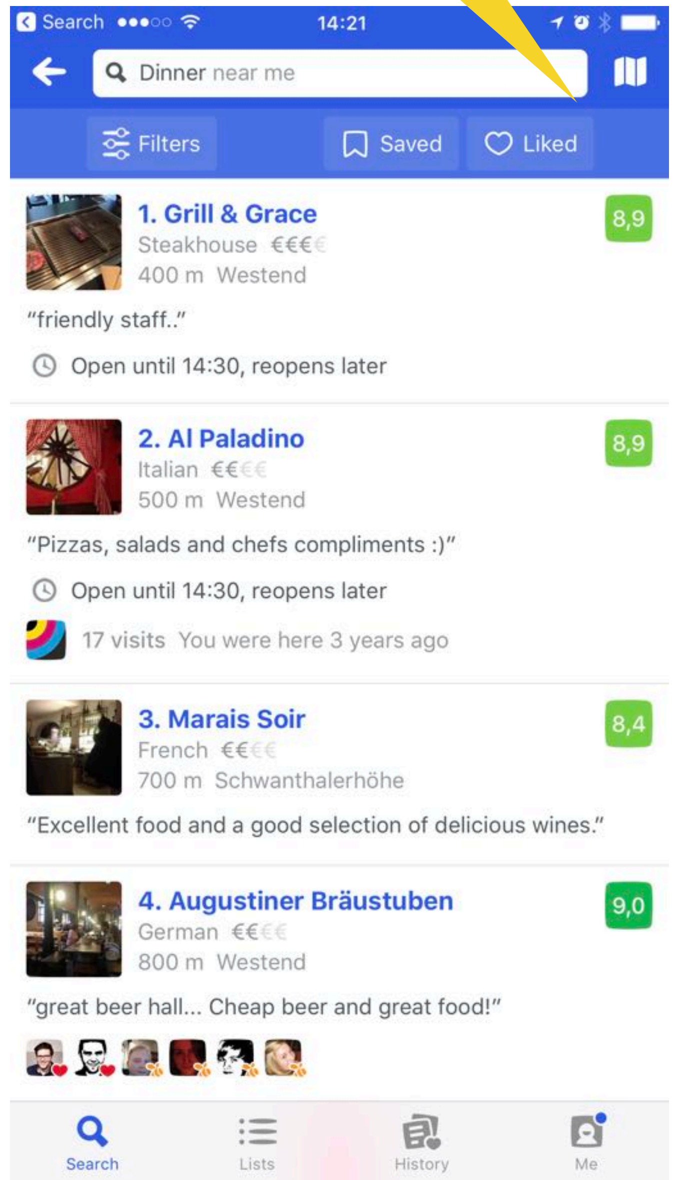


Hey! How do I use Foursquare?

Well, you can use it to check in to places, share your location with friends... Oh, also it's great for finding restaurants nearby and learning more about the bar you're already at...

That's cool, but I don't feel comfortable sharing my location.

Delivered



Search ●●●○○ 14:21

Dinner near me

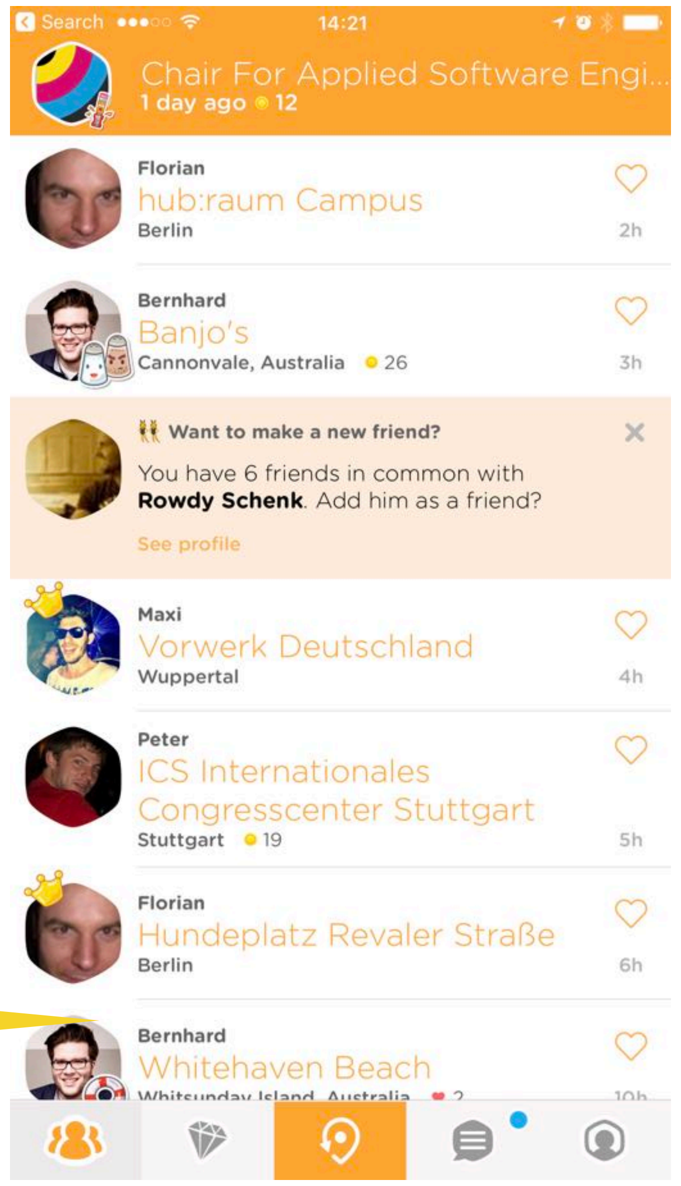
Filters Saved Liked

1. Grill & Grace
Steakhouse €€€€
400 m Westend
8,9
"friendly staff.."
Open until 14:30, reopens later

2. Al Paladino
Italian €€€€
500 m Westend
8,9
"Pizzas, salads and chefs compliments :)"
Open until 14:30, reopens later
17 visits You were here 3 years ago

3. Marais Soir
French €€€€
700 m Schwanthalerhöhe
8,4
"Excellent food and a good selection of delicious wines."

4. Augustiner Bräustuben
German €€€€
800 m Westend
9,0
"great beer hall... Cheap beer and great food!"



Search ●●●○○ 14:21

Chair For Applied Software Engi...
1 day ago ● 12

Florian
hub:raum Campus
Berlin
2h

Bernhard
Banjo's
Cannonvale, Australia ● 26
3h

Want to make a new friend?
You have 6 friends in common with **Rowdy Schenk**. Add him as a friend?
See profile

Maxi
Vorwerk Deutschland
Wuppertal
4h

Peter
ICS Internationales Congresscenter Stuttgart
Stuttgart ● 19
5h

Florian
Hundeplatz Revaler Straße
Berlin
6h

Bernhard
Whitehaven Beach
Whitsunday Island, Australia ♥ 2
10h

Swarm for check-ins

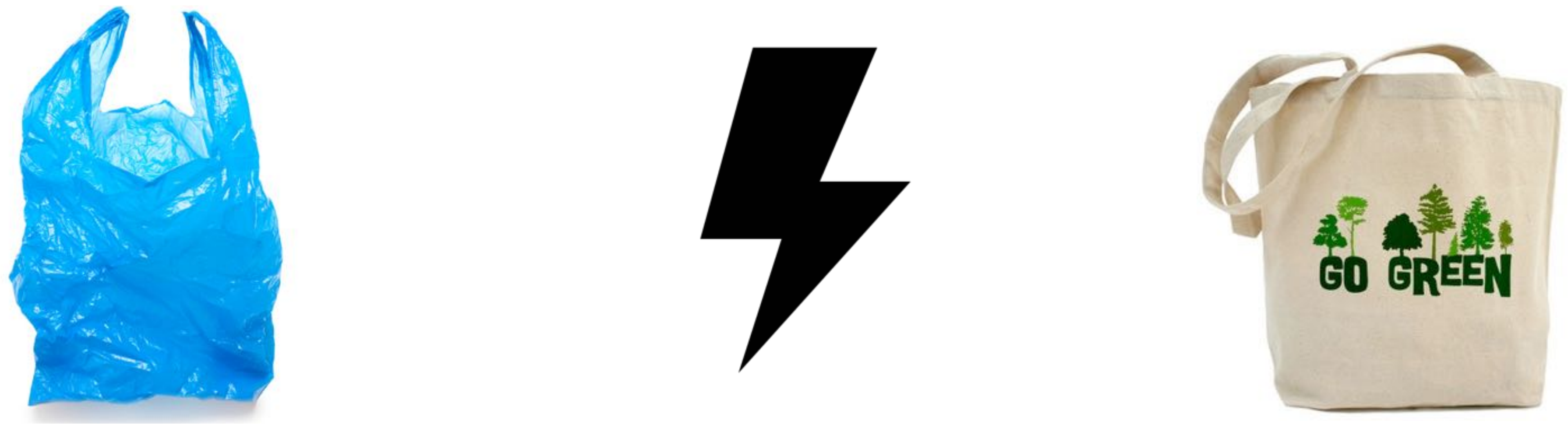
Example: cost vs. reusability

Assume you model the association between 2 classes with an 1-1 multiplicity

- Easy to code, low cost tests, not very reusable

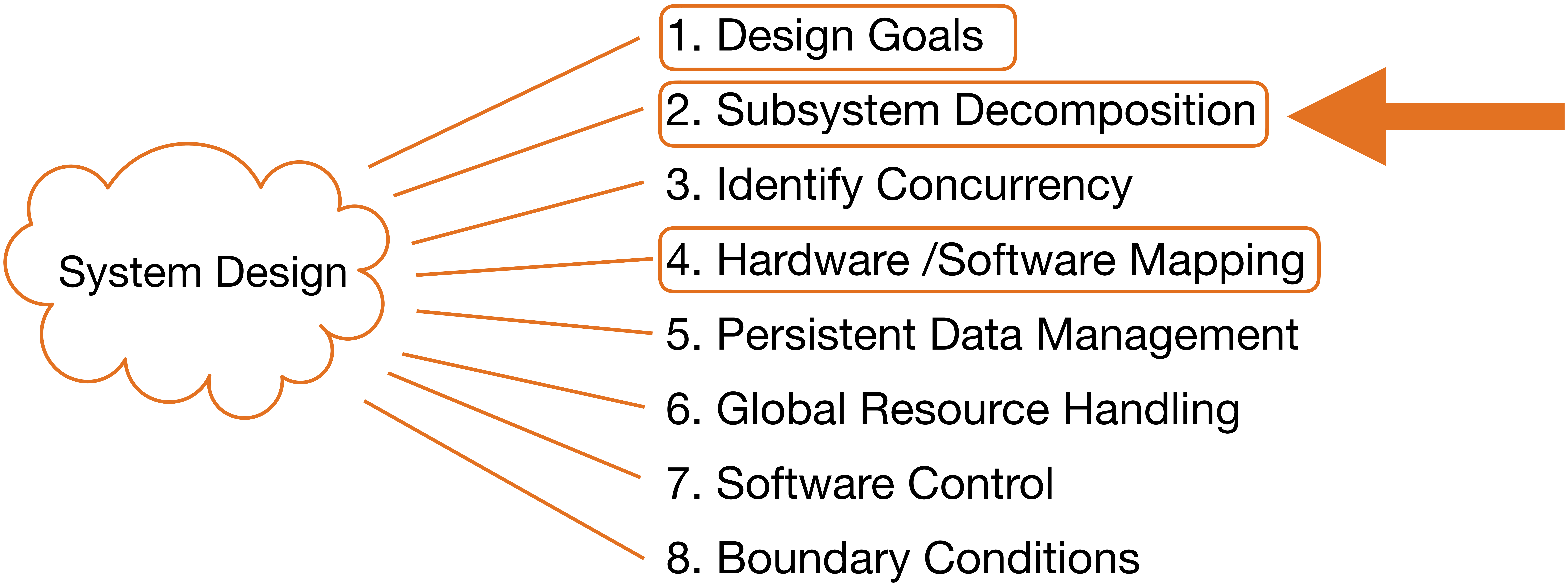
Moving from 1-1 association to a many-many association

- Additional coding and testing costs



With design patterns this trade-off is no longer painful. You can achieve reusability with low cost if you use design patterns.

In this course we cover …



System Design

1. Design Goals
2. Subsystem Decomposition
3. Identify Concurrency
4. Hardware /Software Mapping
5. Persistent Data Management
6. Global Resource Handling
7. Software Control
8. Boundary Conditions

Subsystems and services

Subsystem

- Collection of classes, associations, operations, events that are closely interrelated with each other
- The classes in the object model are the seeds for subsystems

Service

- A group of externally visible operations provided by a subsystem (also called subsystem interface)
- The use cases in the functional model provide the “seeds” for services

Coupling and coherence of subsystems

GOOD SYSTEM DESIGN

Goal: reduce system complexity while allowing change

Coherence (measures dependency among classes)

- High coherence: classes in the subsystem perform similar tasks and are related to each other via many associations
- Low coherence: lots of miscellaneous and auxiliary classes, almost no associations

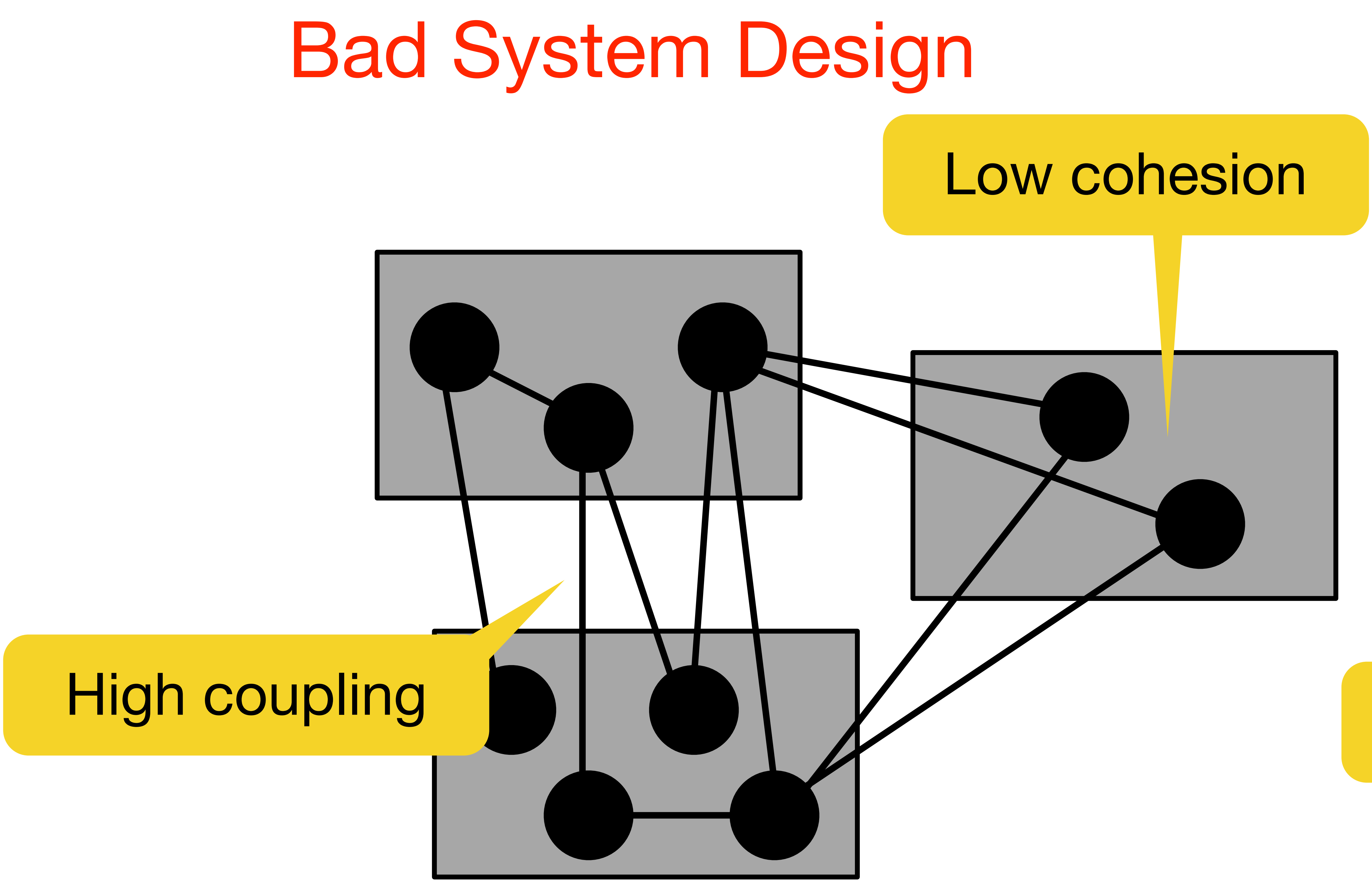
Coupling (measures dependency among subsystems)

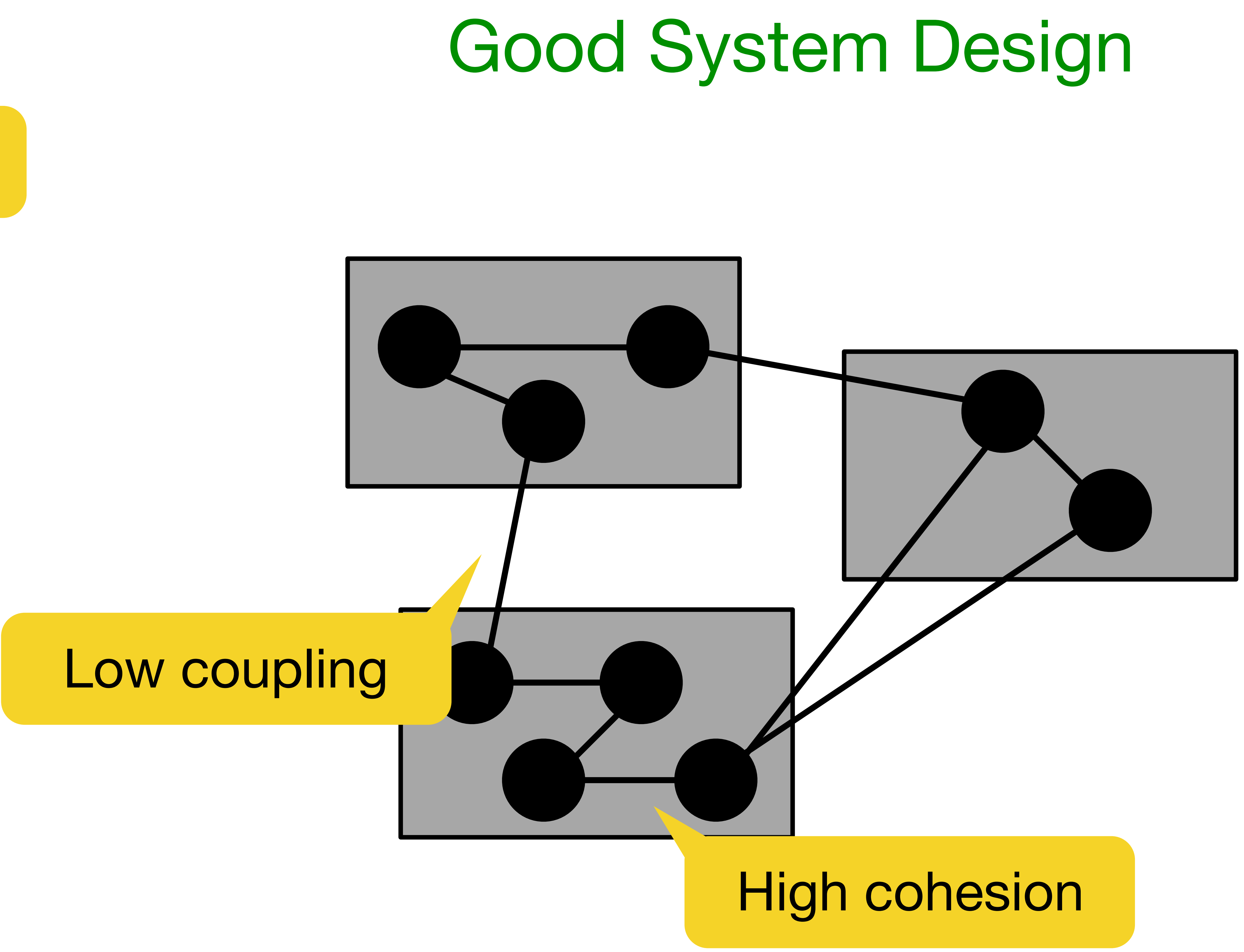
- High coupling: changes to one subsystem will have high impact on other subsystems
- Low coupling: a change in one subsystem does not affect any other subsystem

Cohesion and coupling measure interdependence

Cohesion measures the interdependence of the elements of one subsystem.

Coupling measures the interdependence between different subsystems.





How to achieve high coherence and low coupling?

High coherence can be achieved if most of the interactions is within subsystems, rather than across subsystem boundaries.

Remember: information hiding

Low coupling can be achieved if a calling class does not need to know anything about the internals of the called class.

Architectural style vs. architecture

Subsystem decomposition

Identification of subsystems, services and their relationship to each other

Architectural style

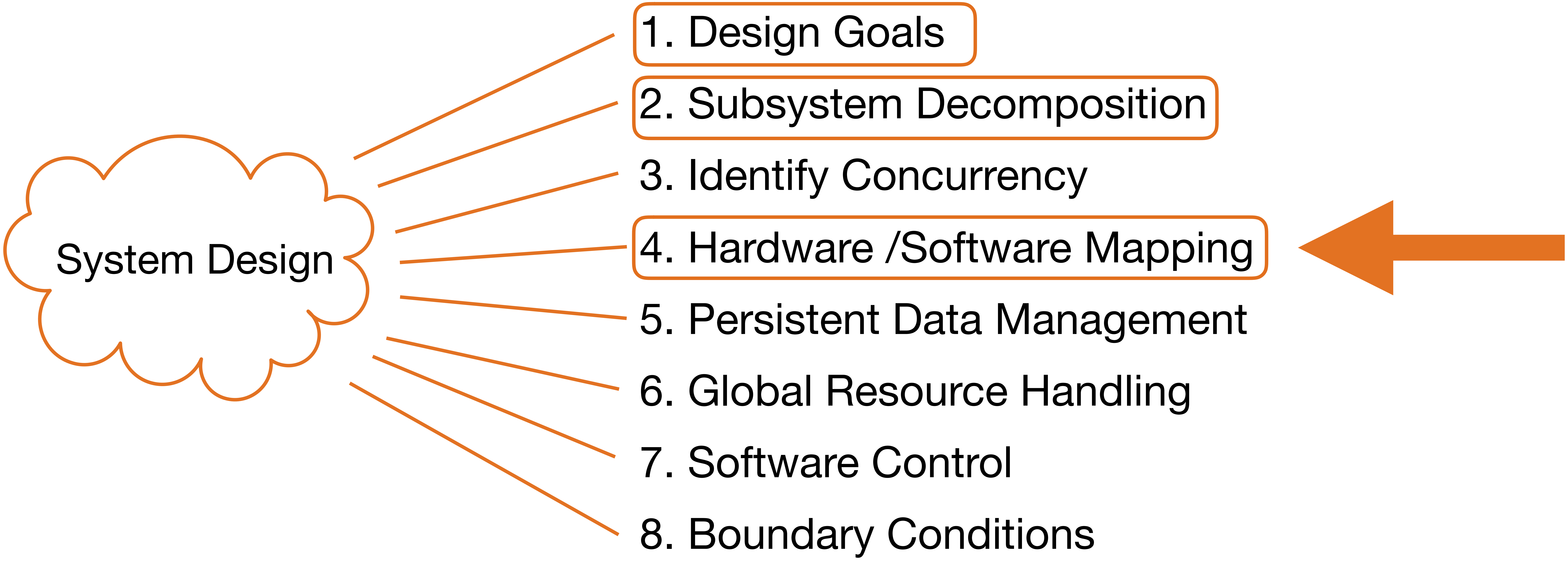
More about this in the unit Architectural Patterns

A pattern for a subsystem decomposition

Software architecture

Instance of an architectural style

In this course we cover …



System Design

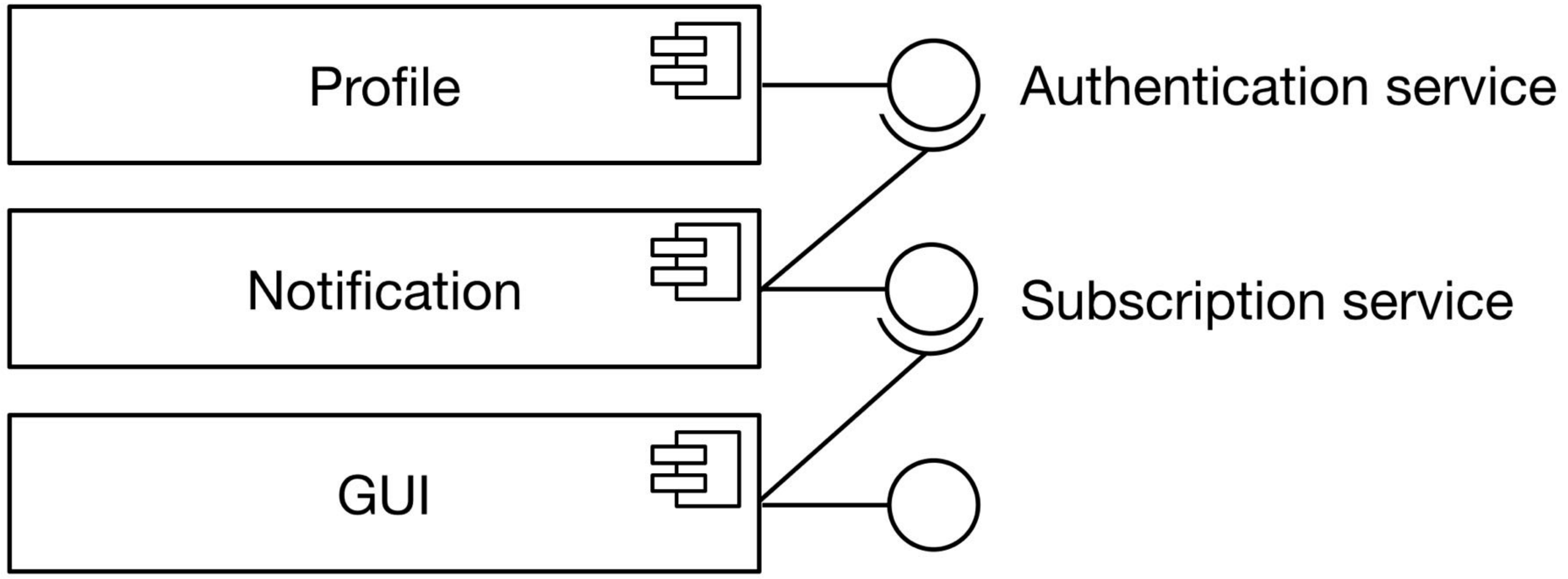
1. Design Goals
2. Subsystem Decomposition
3. Identify Concurrency
4. Hardware /Software Mapping
5. Persistent Data Management
6. Global Resource Handling
7. Software Control
8. Boundary Conditions

Hardware/software mapping

Hardware/software mapping addresses two questions:

1. How shall we realize the subsystems: with hardware or with software?
2. How do we map the object model onto chosen hardware and/or software?

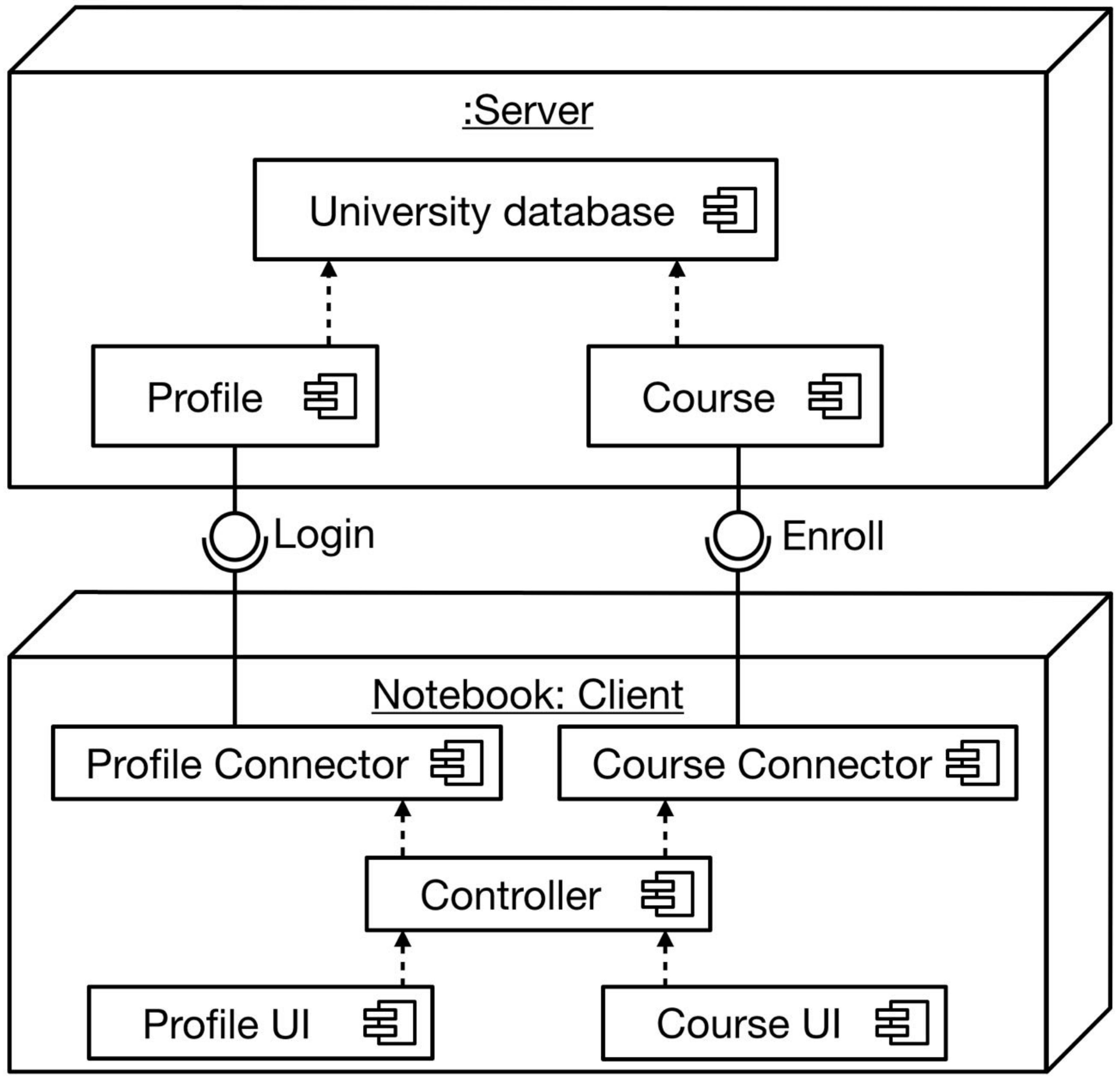
Two UML Diagram Types



Component Diagram

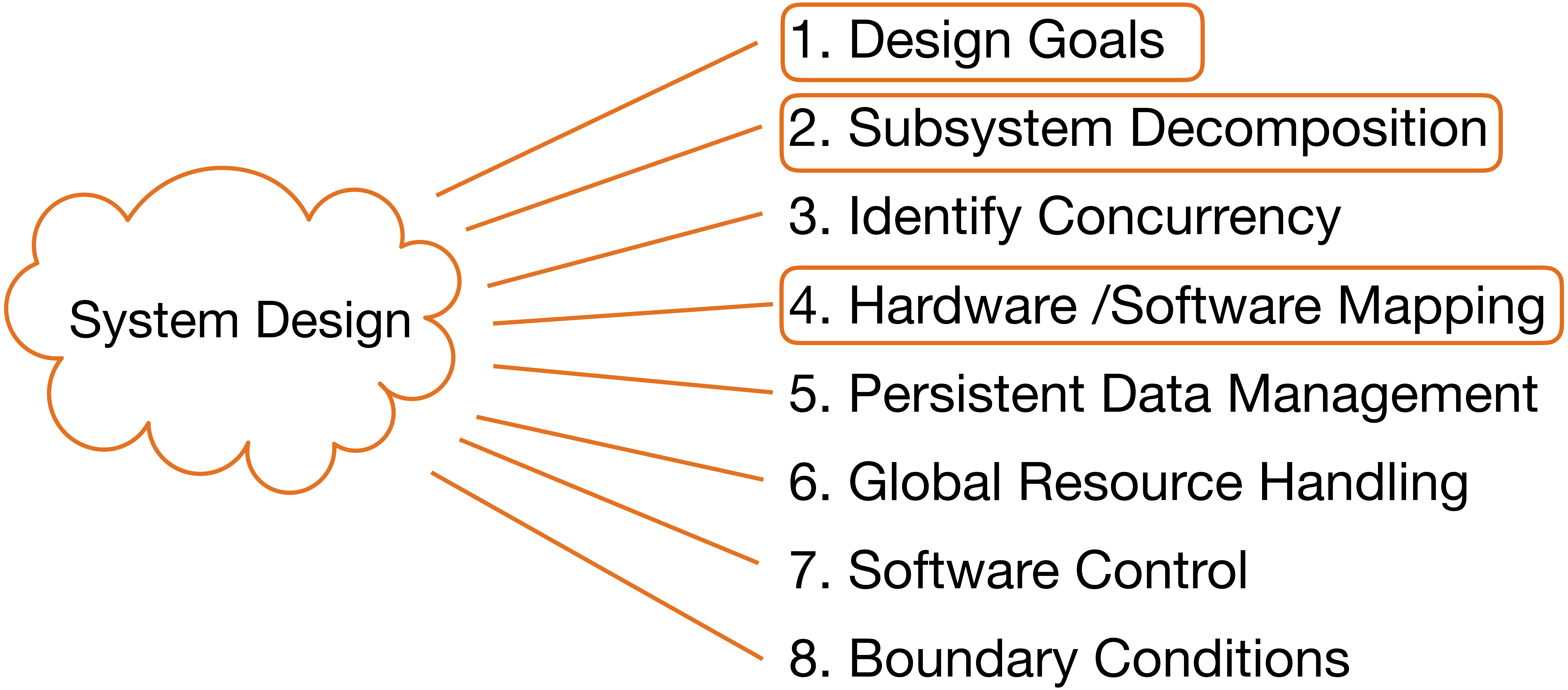
More in the unit “Component Diagram”

More in the unit “Deployment Diagram”



Deployment Diagram

In this course we cover …



System Design

1. Design Goals
2. Subsystem Decomposition
3. Identify Concurrency
4. Hardware /Software Mapping
5. Persistent Data Management
6. Global Resource Handling
7. Software Control
8. Boundary Conditions

Software Engineering Essentials

System Design

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics