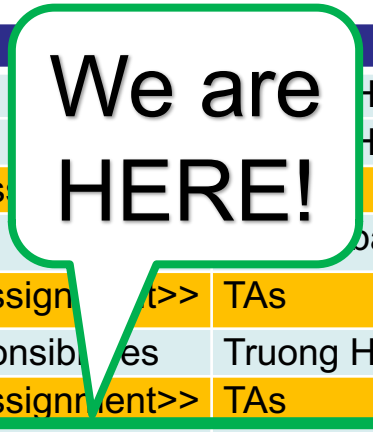
Architectural Styles

Truong Ho-Quang

https://en.wikipedia.org/wiki/De_Stijl

Schedule

Week		Date	Time	Lecture	
36	L1	Wed, 2 Sept	13:15 – 15:00	Introduction & Organization	Ho
37	L2	Wed, 9 Sept	13:15 – 15:00	Architecting Process & Views	Ho
37	S1	Thu, 10 Sept	10:15 – 12:00	<< Supervision/As	
38	L3	Wed, 16 Sept	13:15 - 15:00	Requirements & Quality Attributes	bara
38	S2	Thu, 17 Sept	10:15 – 12:00	<< Supervision/Assign t>>	TAs
38	L4	Fri, 18 Sept	13:15 – 15:00	Architectural Tactics & Roles and Responsib es	Truong Ho
39	S3	Wed, 23 Sept	13:15 – 15:00	<< Supervision/Assignment>>	TAs
39	L5	Thu, 24 Sept	10:15 – 12:00	Functional Decomposition & Architectural Styles P1	Truong Ho
39	L6	Fri, 25 Sept	13:15 – 15:00	Architectural Styles P2	Truong Ho
40	S4	Wed, 30 Sept	13:15 – 15:00	<< Supervision/Assignment>>	TAs
40	L7	Thu, 1 Oct	10:15 – 12:00	Architectural Styles P3	Sam Jobara
40	L8	Fri, 2 Oct	13:00 – 15:00	Guest Lecture: Scaling DevOps – GitHub's Journey from 500+ to 1500+ People	Johannes Nicolai
41	S5	Wed, 7 Oct	13:15 – 15:00	<< Supervision/Assignment>>	TAs
41	L9	Thu, 8 Oct	10:15 – 12:00	Current Industrial SW Architecture Issues: Software Architectures of Blockchain with Case Study	Sam Jobara
42	L10	Wed, 14 Oct	13:15 – 15:00	Design Principles	Truong Ho
42	S6	Thu, 15 Oct	10:15 – 12:00	<< Supervision/Assignment>>	TAs
42	L11	Fri, 16 Oct	13:15 – 15:00	Guest Lecture: Architecture changes at Volvo Truck's Application System (TAS)	Anders Magnusson
43	L12	Wed, 21 Oct	13:15 – 15:00	Architecture Evaluation	Truong Ho
43	L13	Thu, 22 Oct	10:15 – 12:00	Reverse Engineering & Correspondence	Truong Ho
43		Fri, 23 Oct	13:00 – 15:00	To be determined (exam practice?)	Teachers
44	Exam	30 Oct	8:30 – 12:30		



Assignment schedule

Week		Date	Lecture	Assignment 1 – Task 1 (A1T1)	Assignment 1 – Task 2 (A1T2)	Assignment 2 (A2)
36	L1	Wed, 2 Sept	Introduction & Organization			
37	L2	Wed, 9 Sept	Architecting Process & Views	A1T1 released		
37	S1	Thu, 10 Sept	<< Supervision/Assignment>>	Planing A1T1		
38	L3	Wed, 16 Sept	Requirements & Quality Attr.			
38	S2	Thu, 17 Sept	<< Supervision/Assignment>>	Work A1T1		
38	L4	Fri, 18 Sept	Tactics & Roles			
39	S3	Wed, 23 Sept	<< Supervision/Assignment>>	Work A1T1		
39	L5	Thu, 24 Sept	Decomposition & Style P1	Hand-in A1T1		
39	L6	Fri, 25 Sept	Architectural Styles P2		A1T2 released	
40	S4	Wed, 30 Sept	<< Supervision/Assignment>>	Feedback A1T1	Planing A1T2	
40	L7	Thu, 1 Oct	Architectural Styles P3			
40	L8	Fri, 2 Oct	Industrial lecture 1			
41	S5	Wed, 7 Oct	<< Supervision/Assignment>>		Work A1T2	A2 released
41	L9	Thu, 8 Oct	Industrial lecture 2			
42	L10	Wed, 14 Oct	Design Principles			
42	S6	Thu, 15 Oct	<< Supervision/Assignment>>		Work A1T2	Work A2
42	L11	Fri, 16 Oct	Industrial lecture 3		Hand-in A1T2	
43	L12	Wed, 21 Oct	Architecture Evaluation		Feedback A1T2	
43	L13	Thu, 22 Oct	Reverse Engineering			Hand-in A2
43		Fri, 23 Oct	Exam practice			Tue, 27 Oct: Feedback A2
44	Exam	30 Oct				

Assignment schedule

Week		Date	Lecture	Assignment 1 – Task 1 (A1T1)	Assignment 1 – Task 2 (A1T2)	Assignment 2 (A2)
36	L1	Wed, 2 Sept	Introduction & Organization			
37	L2	Wed, 9 Sept	Architecting Process & Views	A1T1 released		
37	S1	Thu, 10 Sept	<< Supervision/Assignment>>	Planing A1T1		
38	L3	Wed, 16 Sept	Requirements & Quality Attr.			
38	S2	Thu, 17 Sept	<< Supervision/Assignment>>	Work A1T1		
38	L4	Fri, 18 Sept	Tactics & Roles			
39	S3	Wed, 23 Sept	<< Supervision/Assignment>>	Work A1T1		
39	L5	Thu, 24 Sept	Decomposition & Style P1	Hand-in A1T1		
39	L6	Fri, 25 Sept	Architectural Styles P2		A1T2 released	
40	S4	Wed, 30 Sept	<< Supervision/Ass…nent>>	Feedback A1T1	Planing A1T2	
40	L7	Thu, 1 Oct	Architectural Style…			
40	L8	Fri, 2 Oct				
41	S5	Wed, 7 Oc…	…>		Work A1T2	A2 released
41	L9	Thu, 8 Oct…				
42	L10	Wed, 14 C…				
42	S6	Thu, 15 Oc…	…>		Work A1T2	Work A2
42	L11	Fri, 16 Oct	Industrial lecture 3		Hand-in A1T2	
43	L12	Wed, 21 Oct	Architecture Evaluation		Feedback A1T2	
43	L13	Thu, 22 Oct	Reverse Engineering			Hand-in A2
43		Fri, 23 Oct	Exam practice			Tue, 27 Oct: Feedback A2
44	Exam	30 Oct				

Hand-in TODAY!

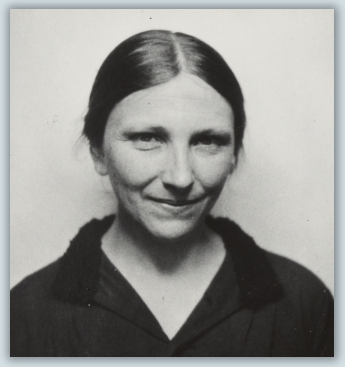
Voluntary student representatives

Drop me an email.

Architectural Styles

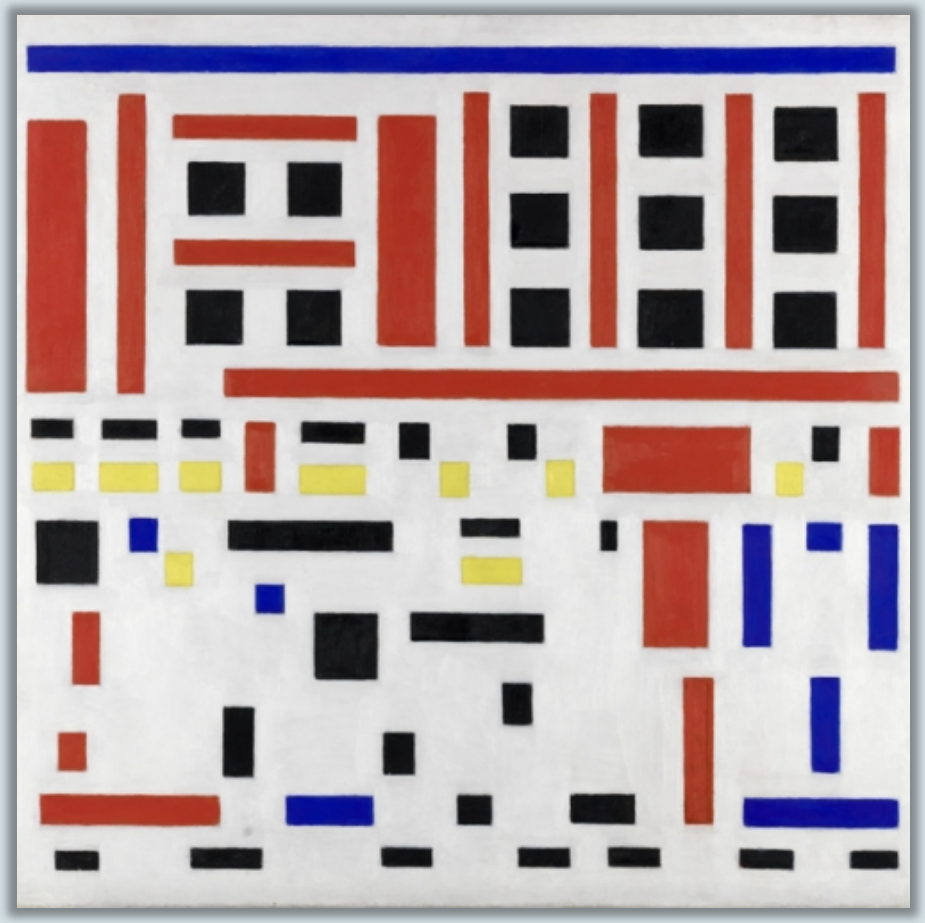
Truong Ho-Quang

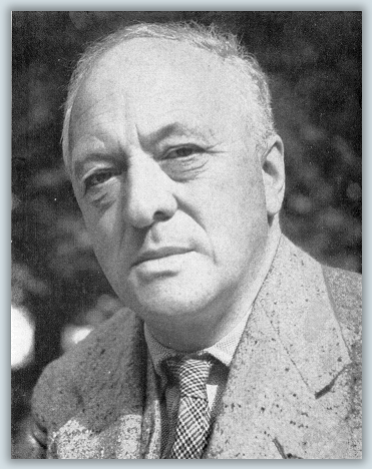
Piet Mondriaan



Truus Schröder-Schräder

Gerrit Rietveld





Bart van der Leck

(1876 – 1958)

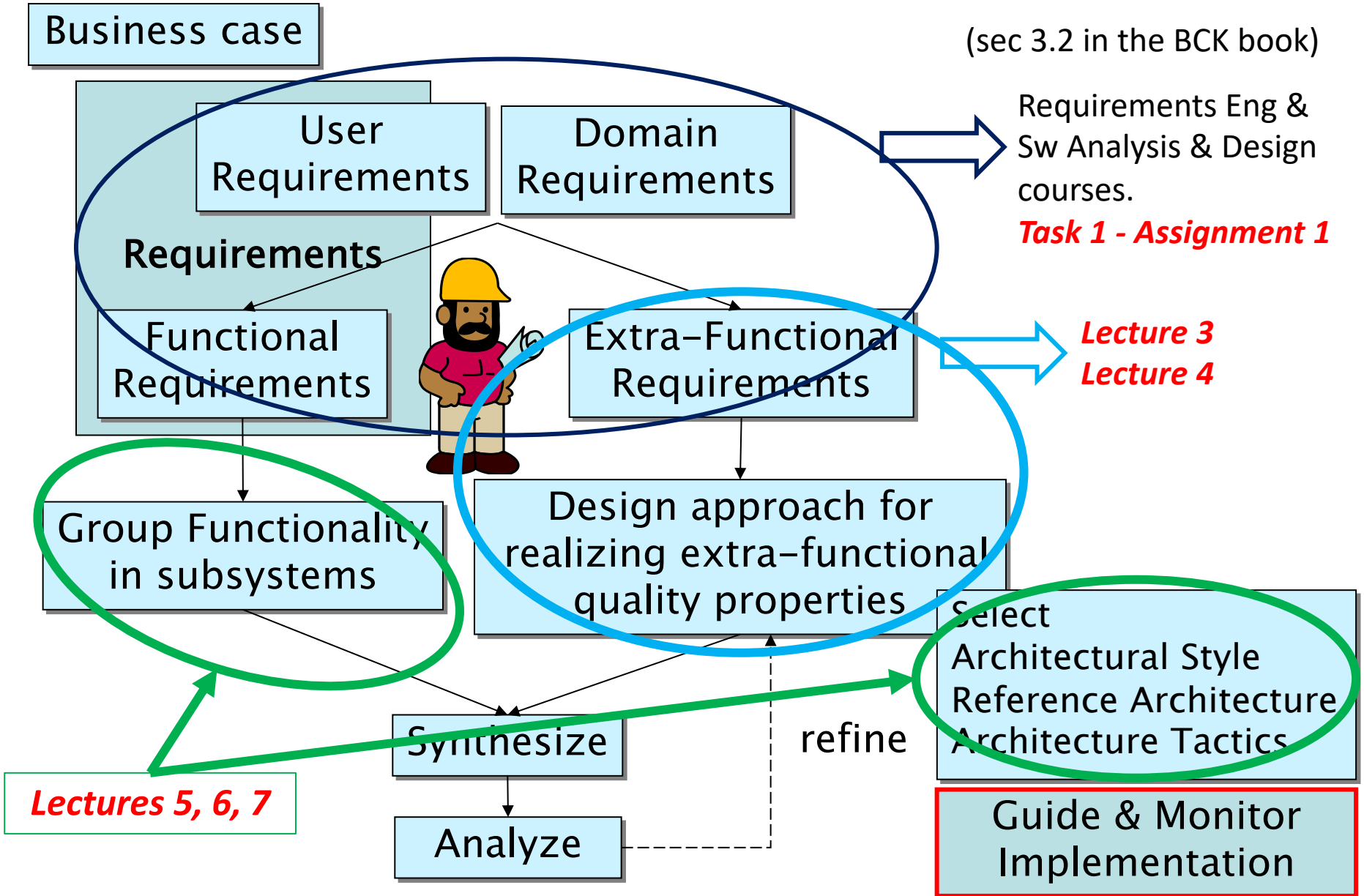
[Composition no. 4, Uitgaan van de fabriek, 1917]

Which ‘Rules’ Define this style?

Recap

- *Forces* influence the design of the architecture
- *Architectural drivers*/*quality attributes* are the forces that dominate the architecture design decisions
- *Architectural tactics* are the design decisions that influences the achievement of quality attribute responses
- Notion of *roles, responsibilities, and collaboration*.

Where we are in the SW Arch Design Process?



Learning Objectives of this lecture

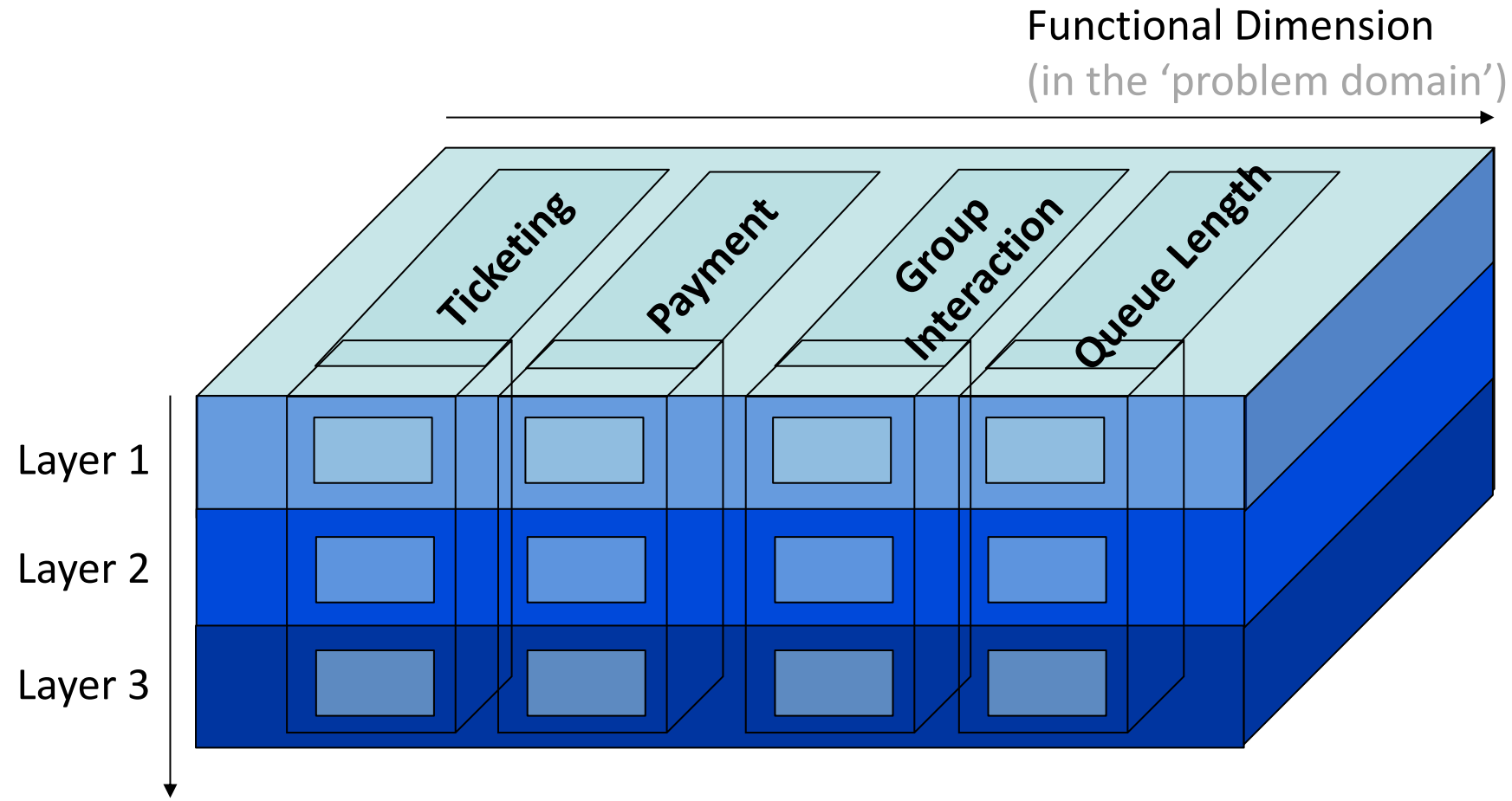
The task of the architect is to come up with a good metaphor for the system
Alexander Ran (Nokia)

- Understand functional decomposition as design approach
- Build vocabulary of architectural styles
  - a set of 'archetypes' that are often used
  - know their relative strengths and weaknesses
- Today: (at least) Client-Server
- Know when to apply or *not* to apply a particular style

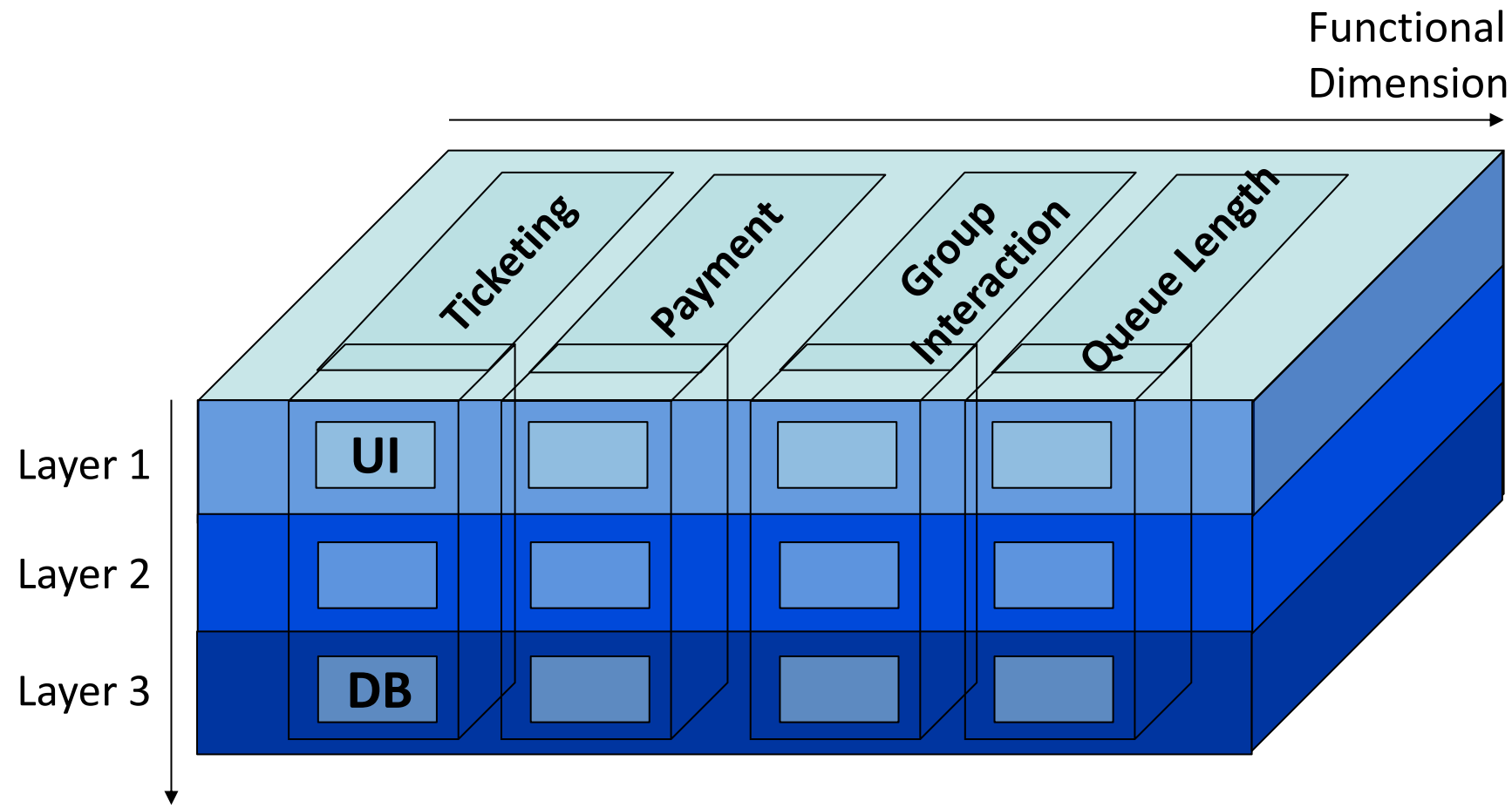
Decomposition of Functionality

- Functional decomposition answer the question: "What are the functions this software must provide?"
- Decomposing is needed to define fine-grain functions
- Functional requirements documents (FD) is a textual representation of functional decomposition. This can be used:
  - as the first step of development
  - as a base of contract with stakeholders

Subsystems vs Layering



Subsystems vs Layering



Radio-Alarm Clock

Identify from subsystems the radio-alarm clock can be built.

Take 4 minutes to write down your ideas

Share your ideas via:

- Zoom Chat (good with sharing text)
- Slack channel "**interactive_lecture**" (photo, screen capture)

Tips: Think of roles/responsibilities

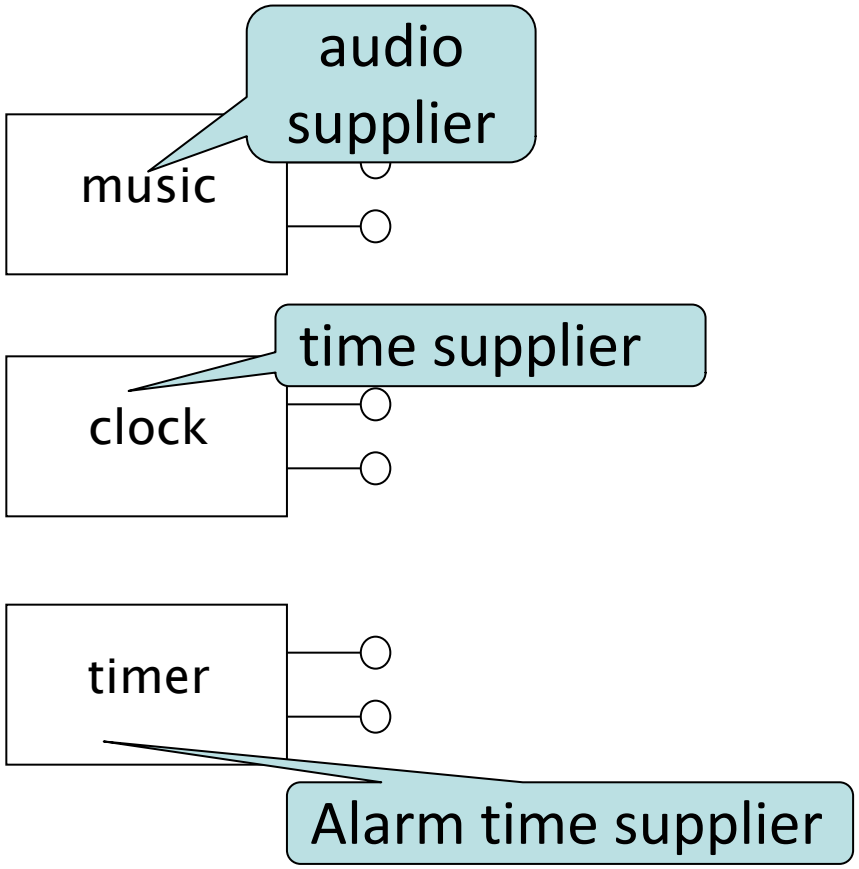
Identify from **subsystems** the radio-alarm clock can be built?

Radio Alarm Clock (initial)

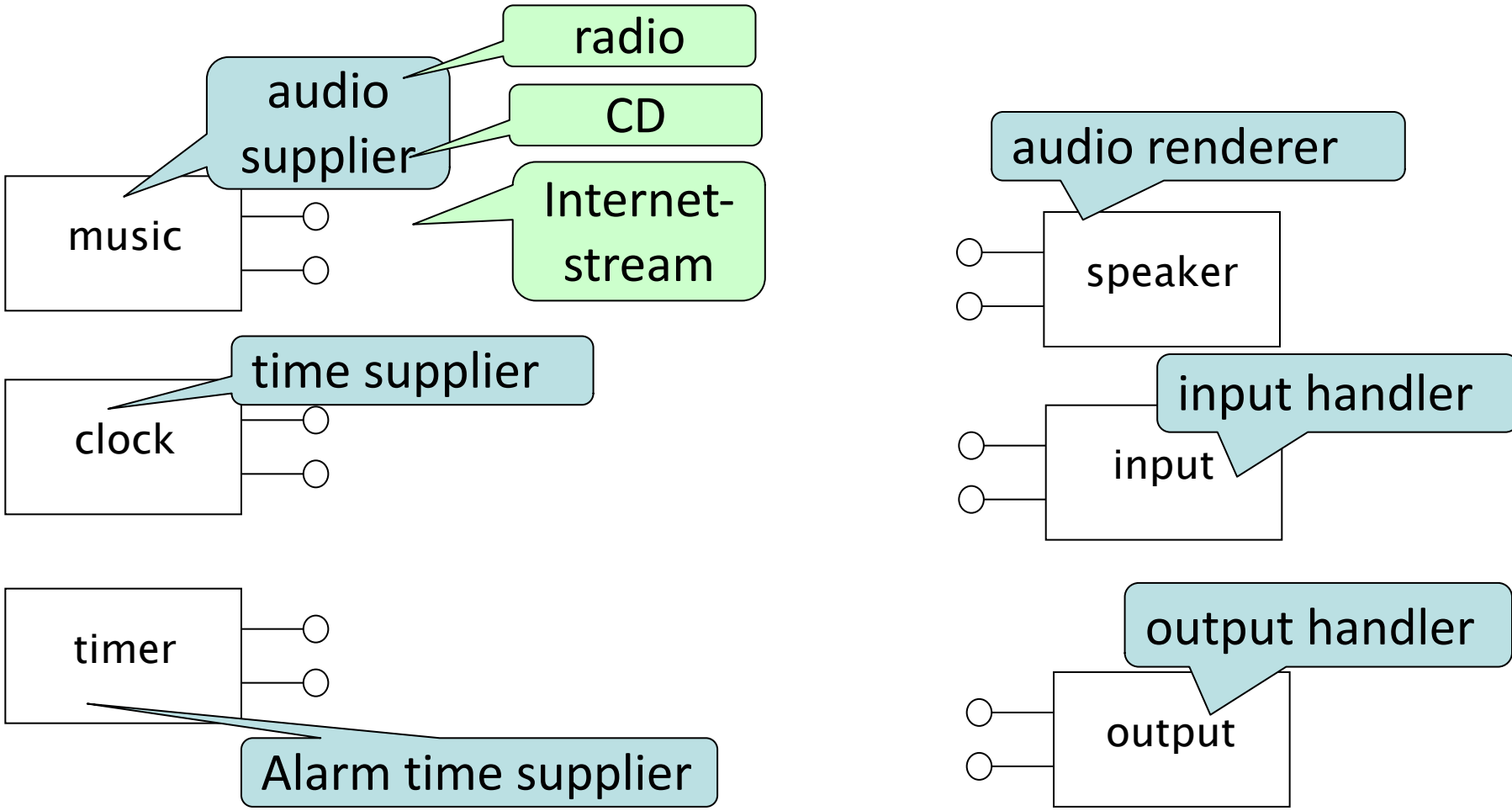
What should be the responsibility of each component?

Radio Alarm Clock

Naming: aim for generality

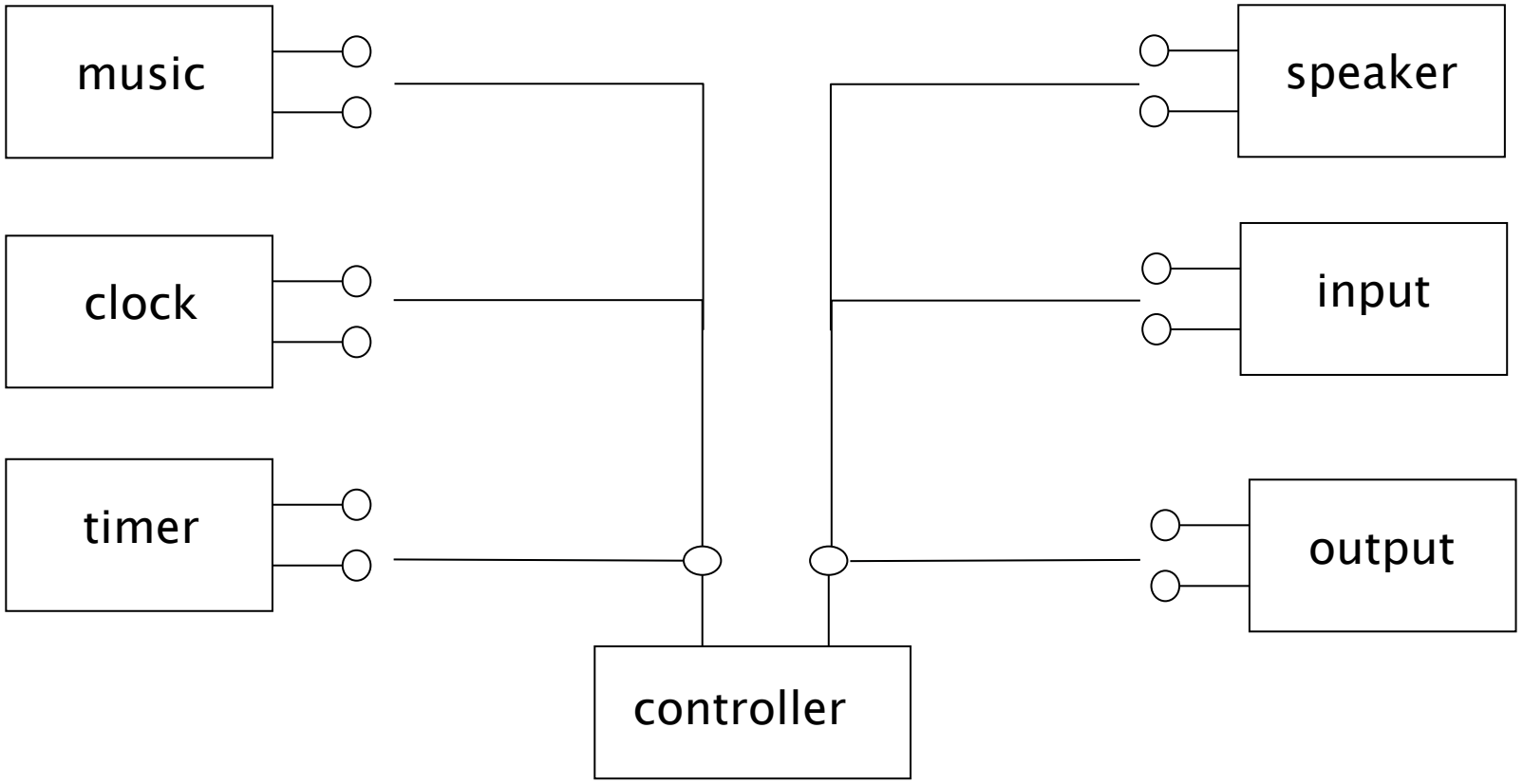


Radio Alarm Clock

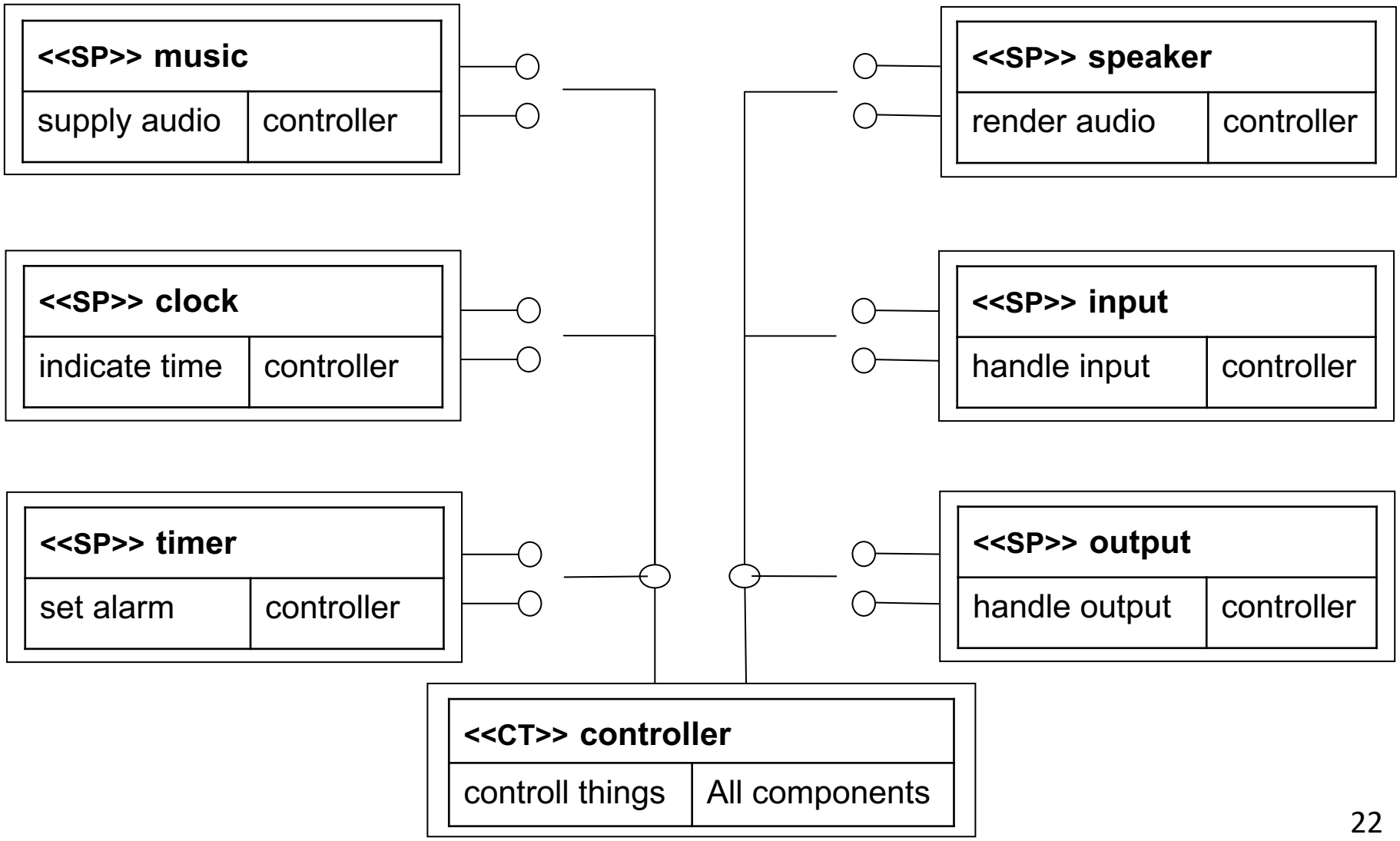


Radio Alarm Clock

A ‘controller’ is an ‘integrator’ of all functionalities



Radio Alarm Clock (with CRC cards)



Radio Alarm Clock

Can your design easily accommodate extensions?

lamp

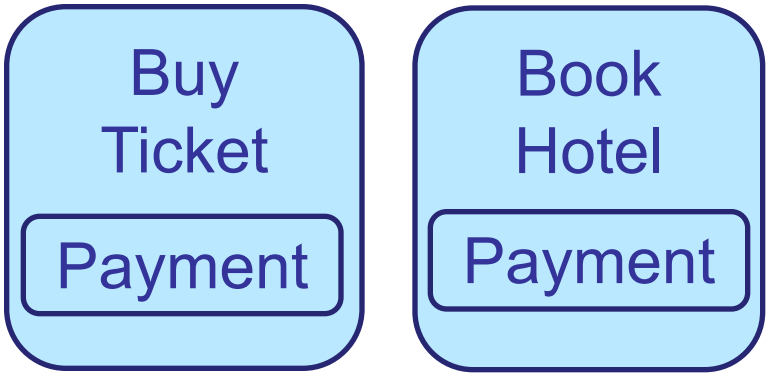
temperature

Bat-alarm

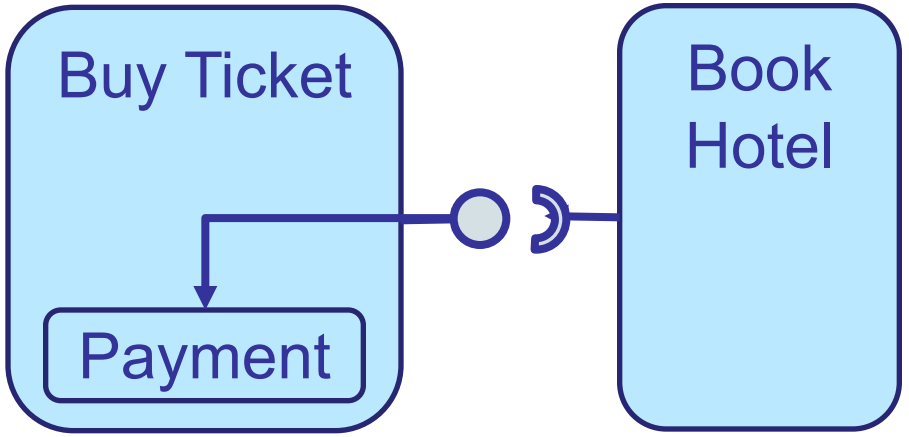
(wireless) atomic clock

Train strike/traffic delays

Which Design and Why?

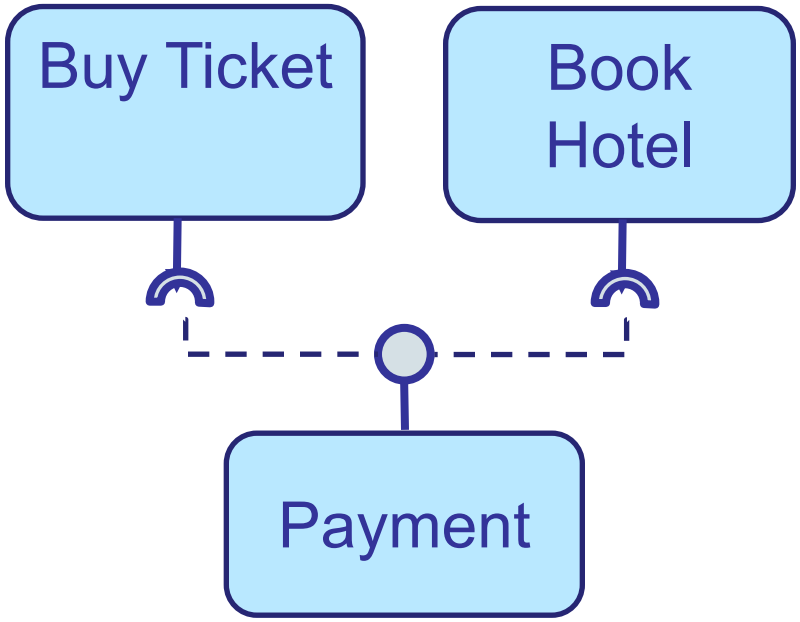


A



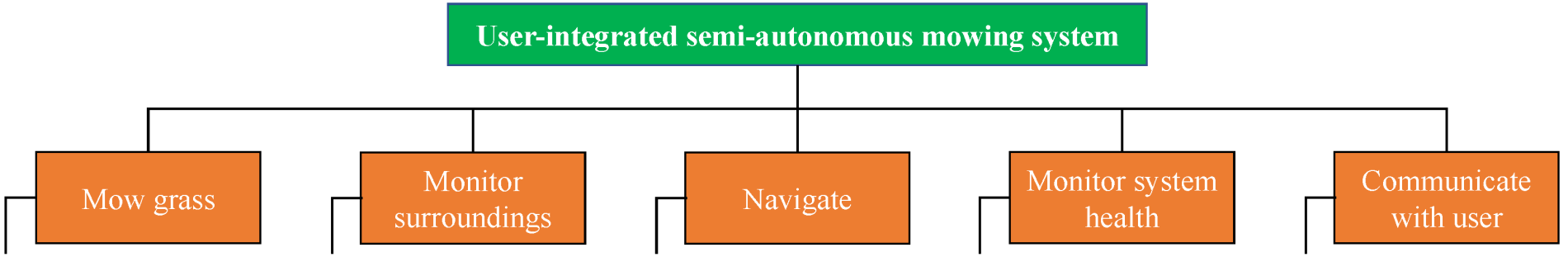
B

Factor out what is common



Payment-functionality is

1) a common, generic service
2) a clear cohesive responsibility
3) a unit of change



Shows:

- Decomposition into main functions of the system

Alt: responsibilities / tasks

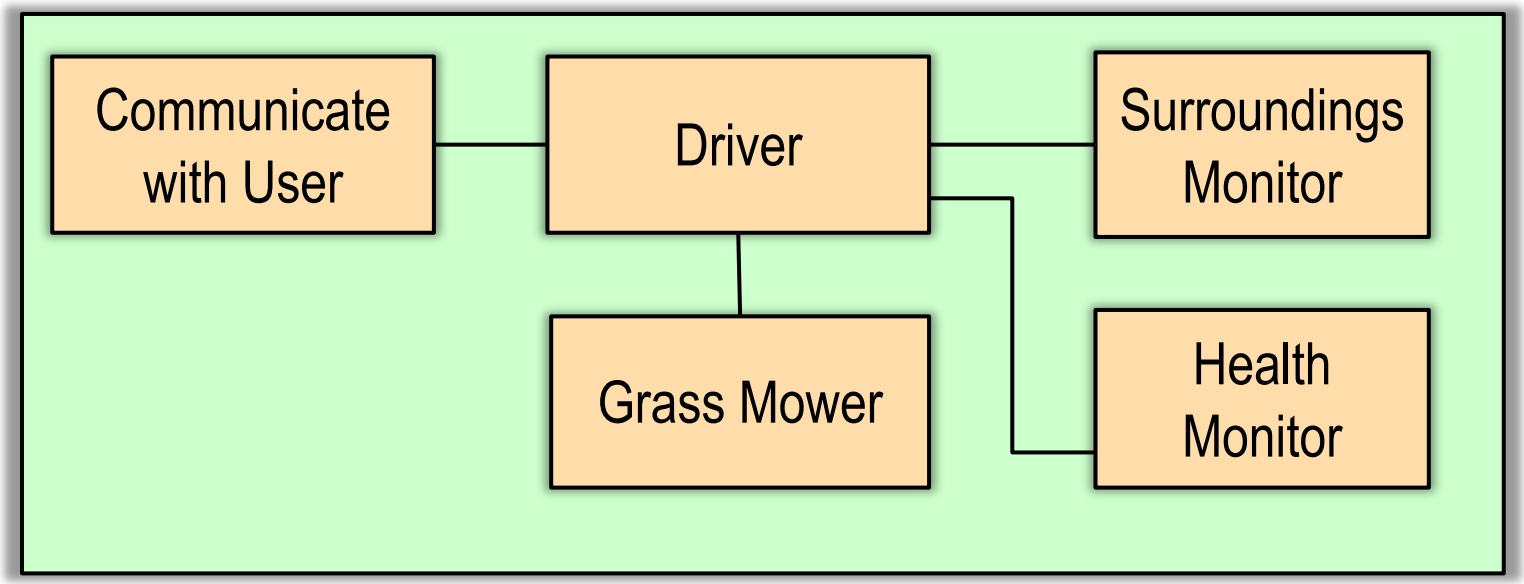
Does not show:

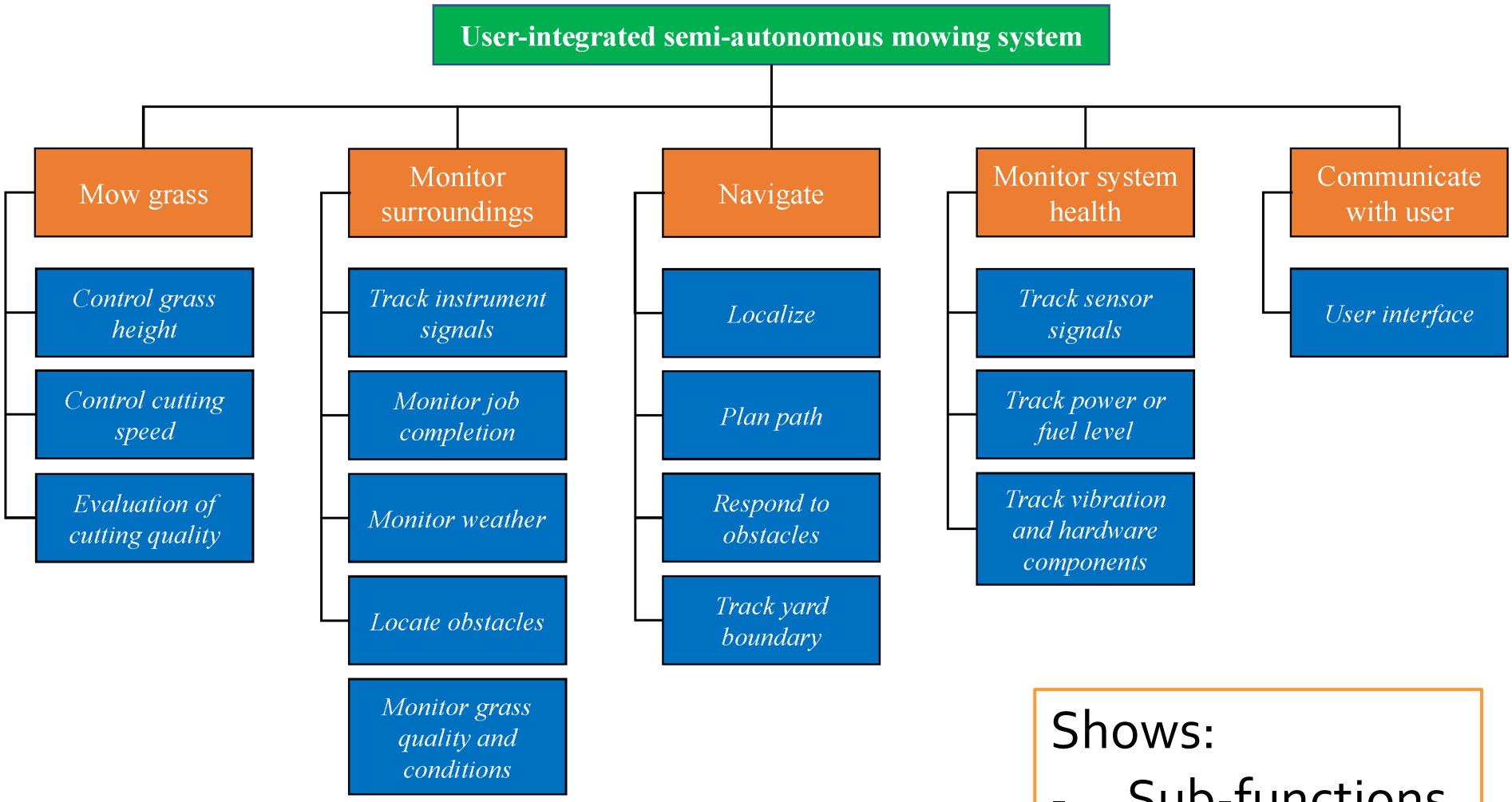
- How components are implemented
- IT does not show 'power' or 'memory'!

Not at this 'perspective'/abstraction

AgriEngineering 2019, 1(3), 453–474; https://doi.org/10.3390/agriengineering1030033
Development of User-Integrated Semi-Autonomous Lawn Mowing Systems: A Systems Engineering Perspective and Proposed Architecture by Albert E. Patterson,Yang Yuan and William R. Norris

Autonomous Grass Mower
Subsystem decomposition

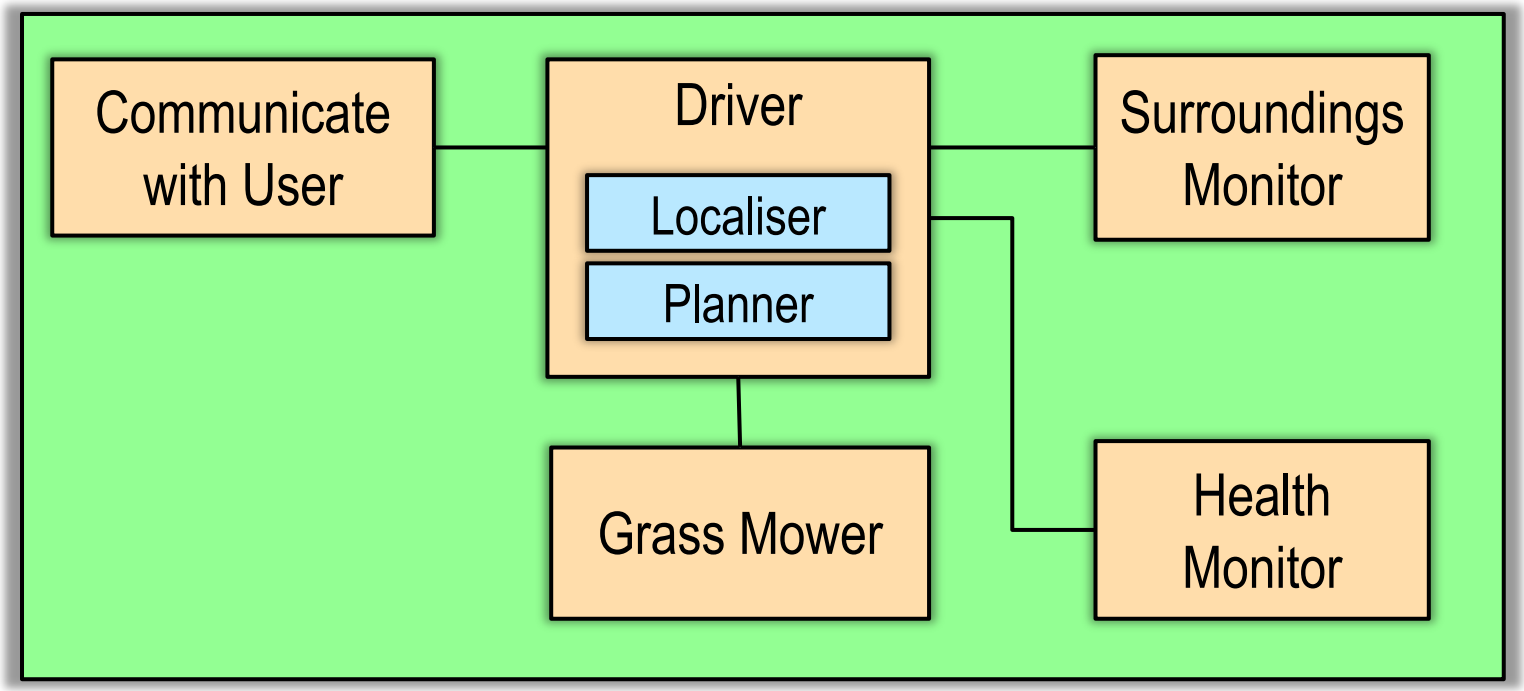




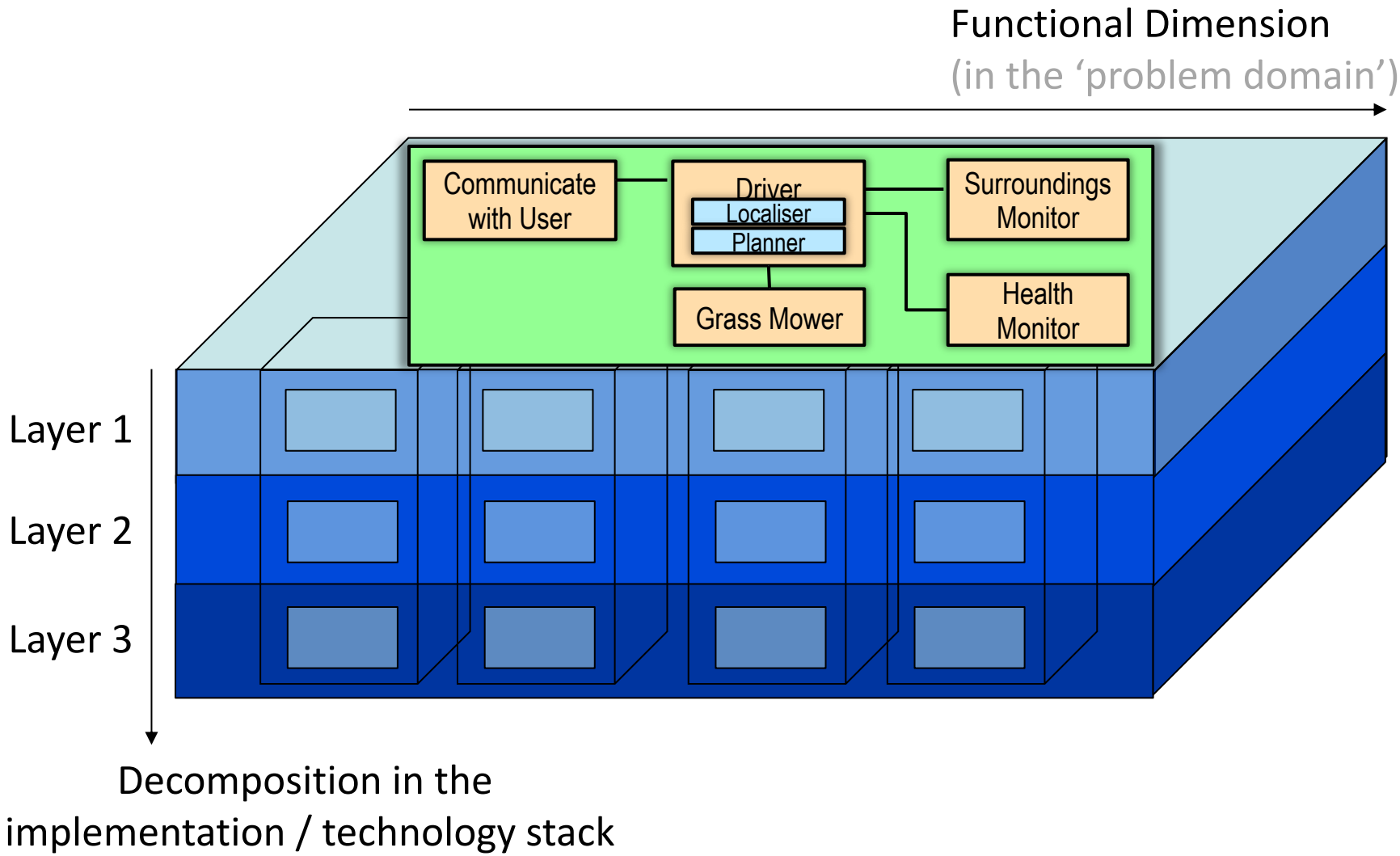
Shows:
- Sub-functions

AgriEngineering 2019, 1(3), 453-474; https://doi.org/10.3390/agriengineering1030033
Development of User-Integrated Semi-Autonomous Lawn Mowing Systems: A Systems Engineering Perspective and Proposed Architecture by Albert E. Patterson,Yang Yuan and William R. Norris

Autonomous Grass Mower
Sub-subsystem decomposition

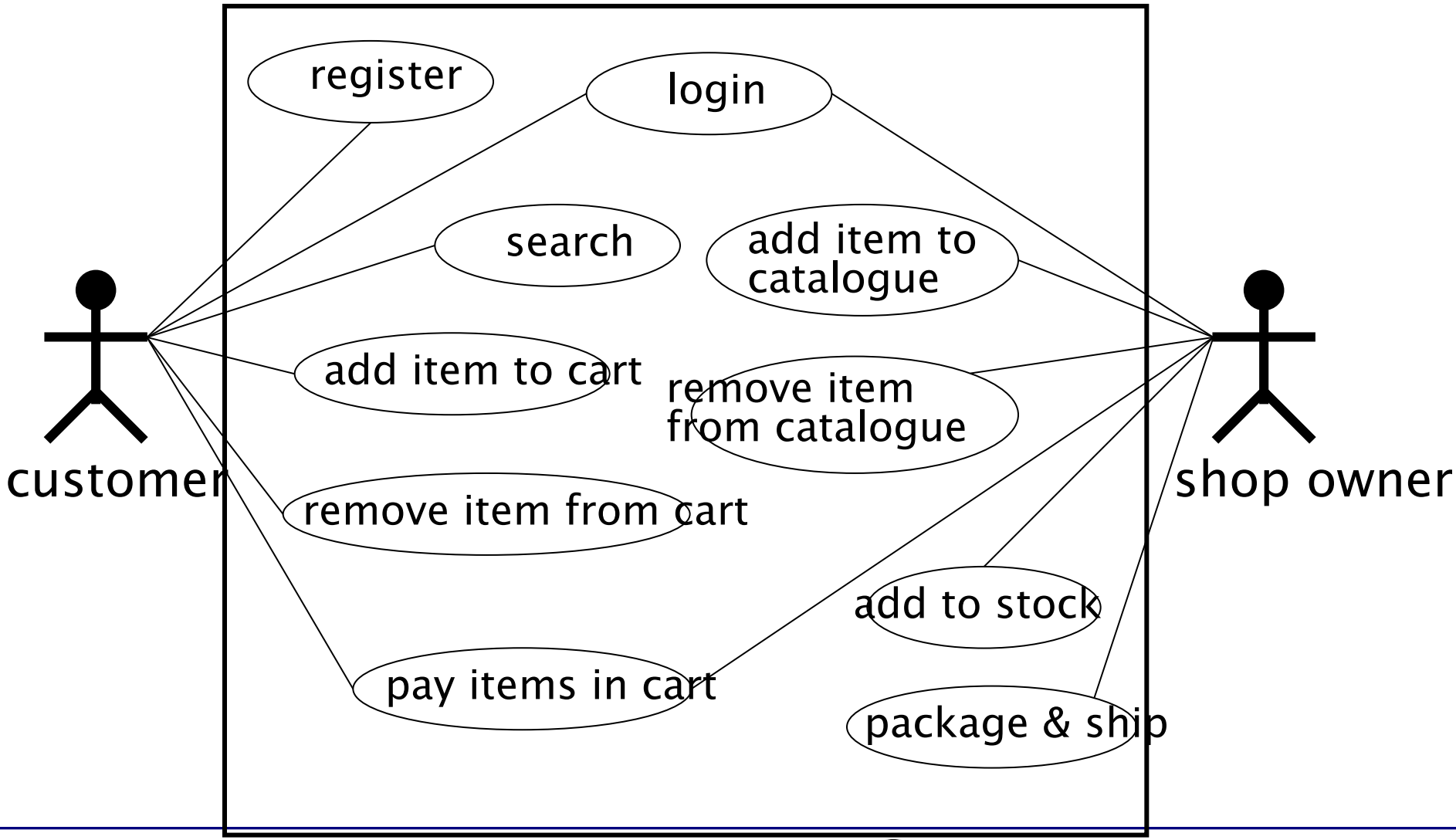


Subsystems vs Layering



Example Design Case: Web Shop

Example Design Case: Web Shop

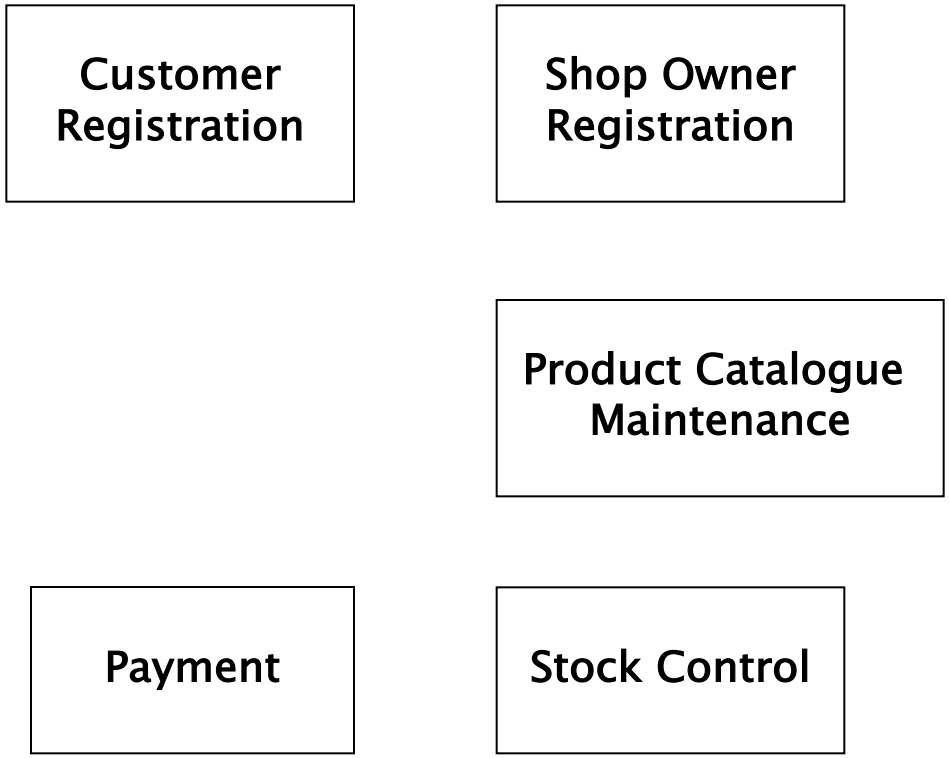


Structure/Group Functionality

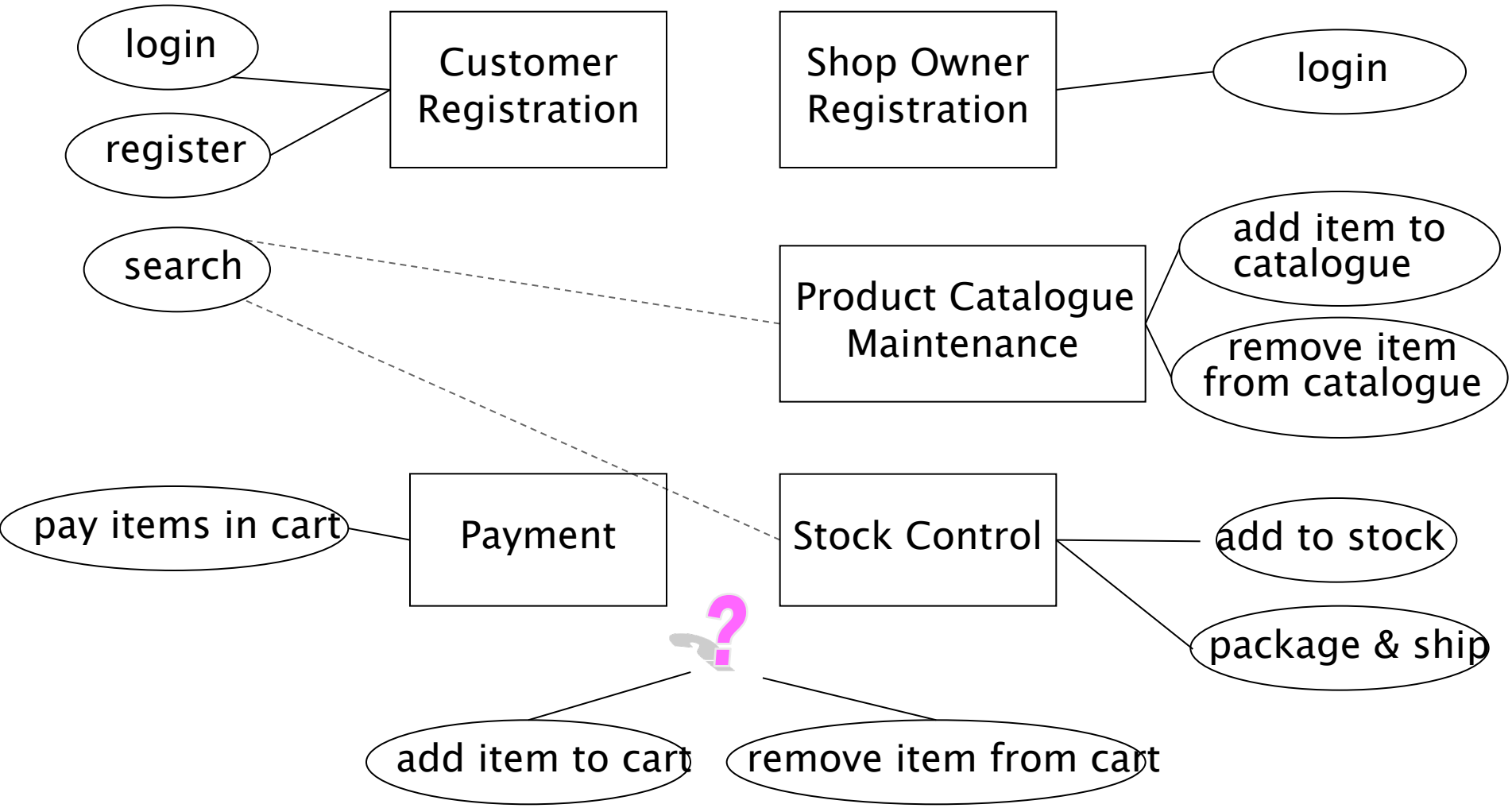
- Defines subsystems of functionality
- Purpose
  - Define decomposition into subsystems
  - Provide support for use-cases
- Think in terms of responsibilities

- Use Component diagram

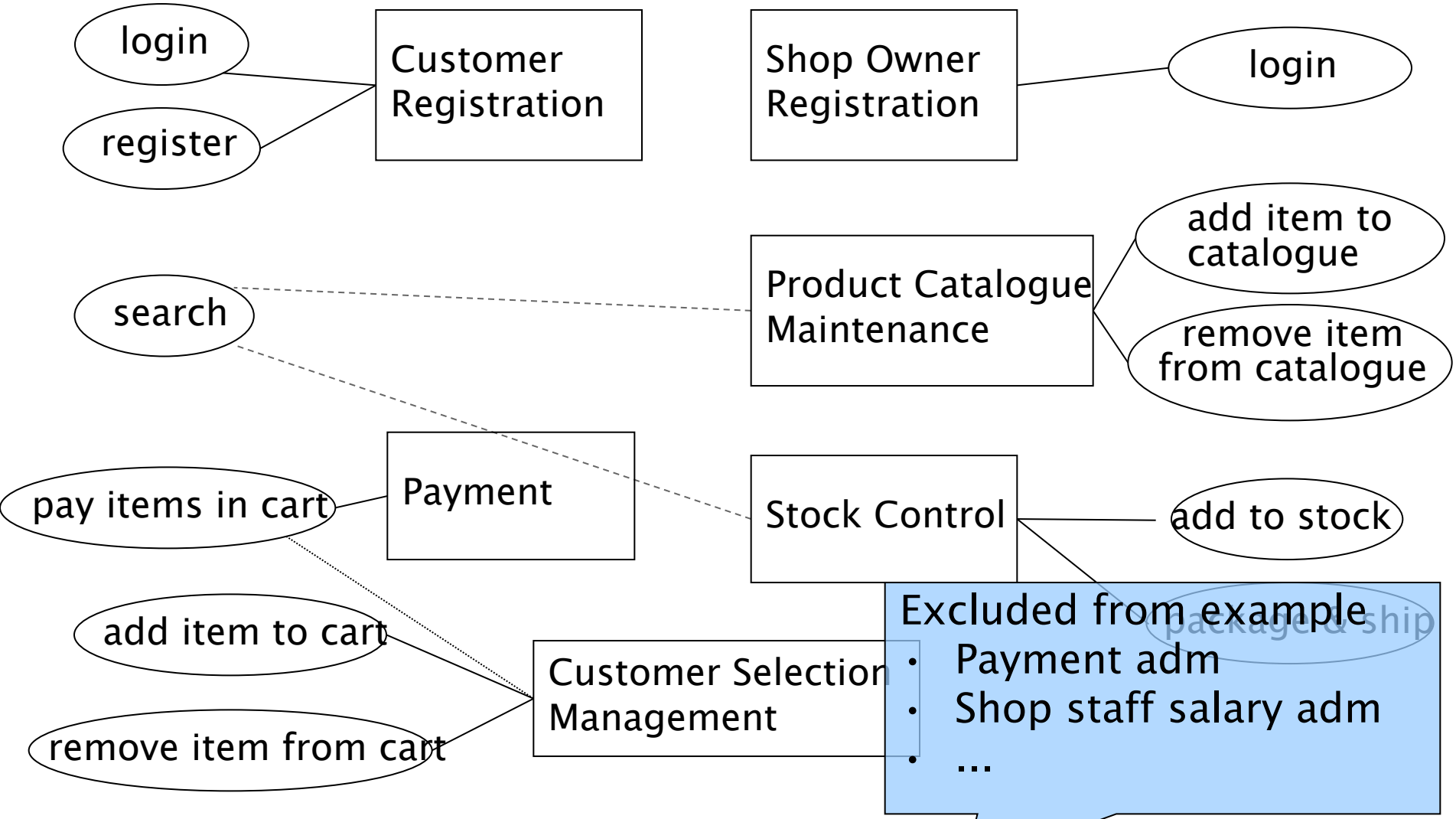
Web Shop: Functional Areas (V0.1)



Check Use Cases Against Functional Areas



Web Shop: Functional Areas (V0.2)



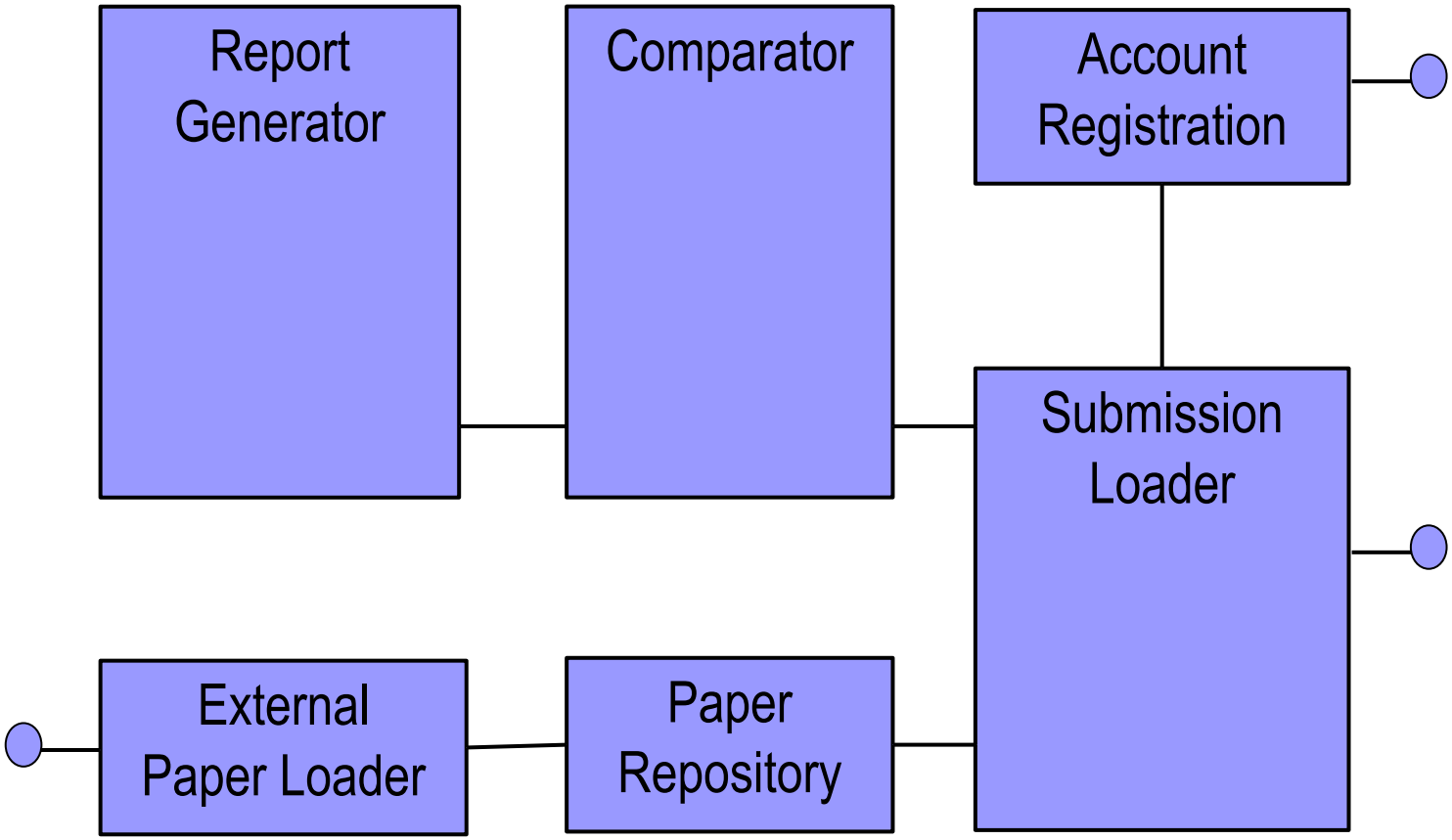
Web Shop: Responsabilities

Customer Registration	Maintain customer accounts
Shop Owner Registration	Maintain staff accounts
Product Catalog Maint.	Maintain product data
Cust. Selection Mngmt.	Maintain customer product selection
Payment	Handle payment between customer, shop & bank
Stock Control	Maintain availability of products in stock

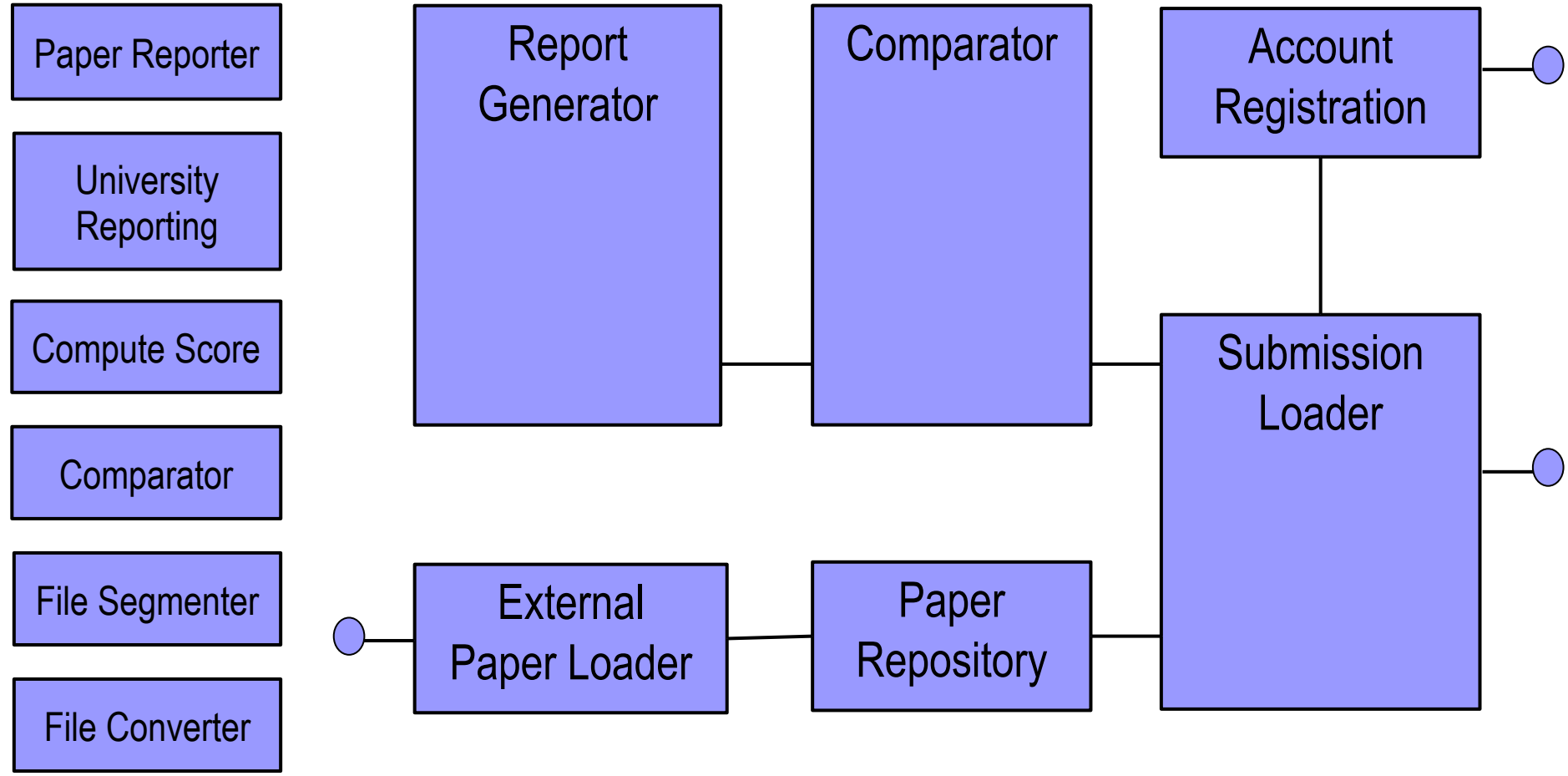
Another Example: Automated Plagiarism Checking System

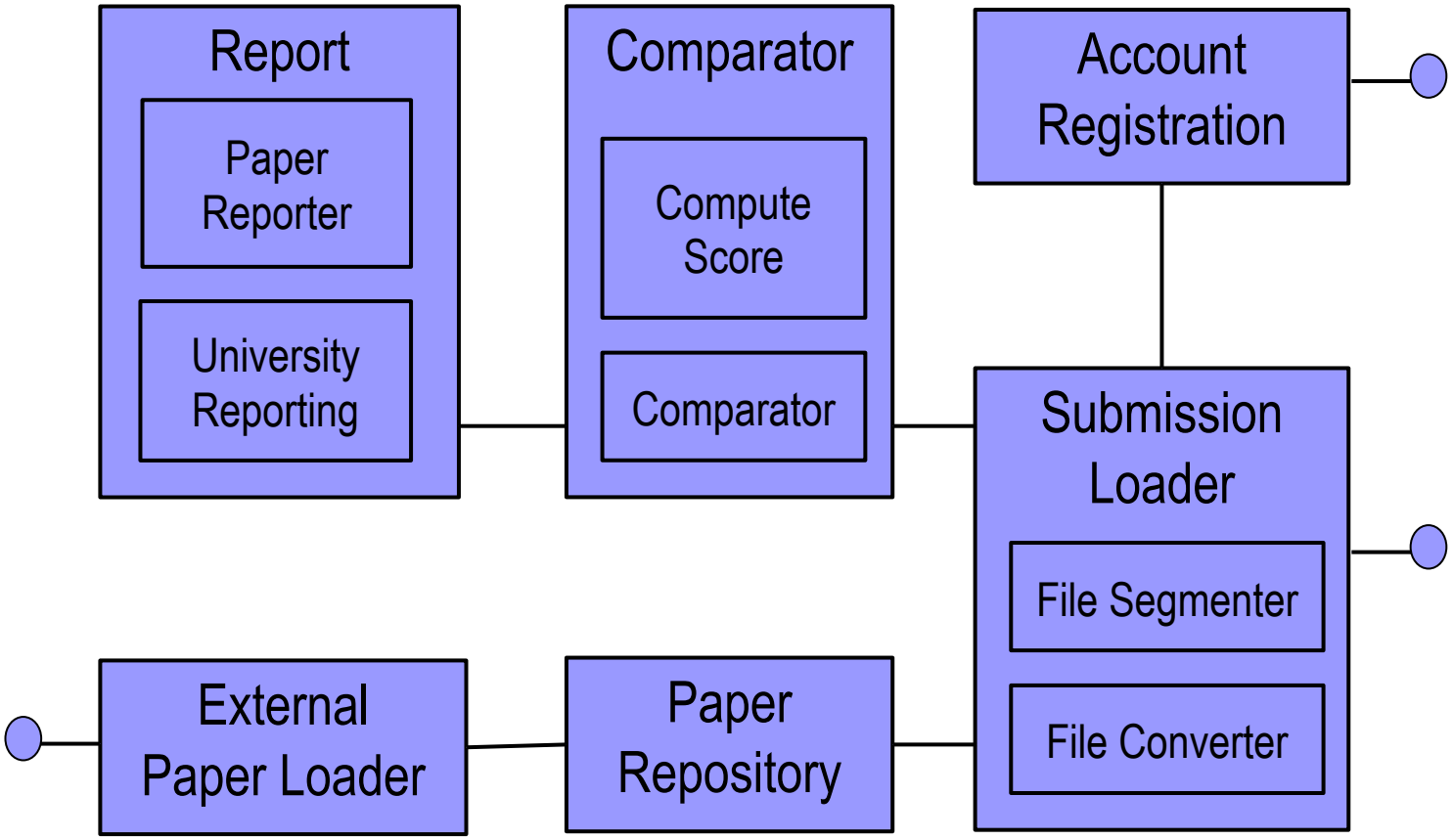
- University can have subscriptions
- University-faculty can make accounts
- Faculty can send in documents for checking
  - Documents are turned into a standard internal format
  - The document is segmented (chapters, section, sentences, …)
  - Document is compared on a sentence by sentence basis.
  - A plagiarism score is produced
  - A report is sent to the person that sent in the document
- The system keeps records of use for producing yearly accounting reports

Decomposition into Subsystems

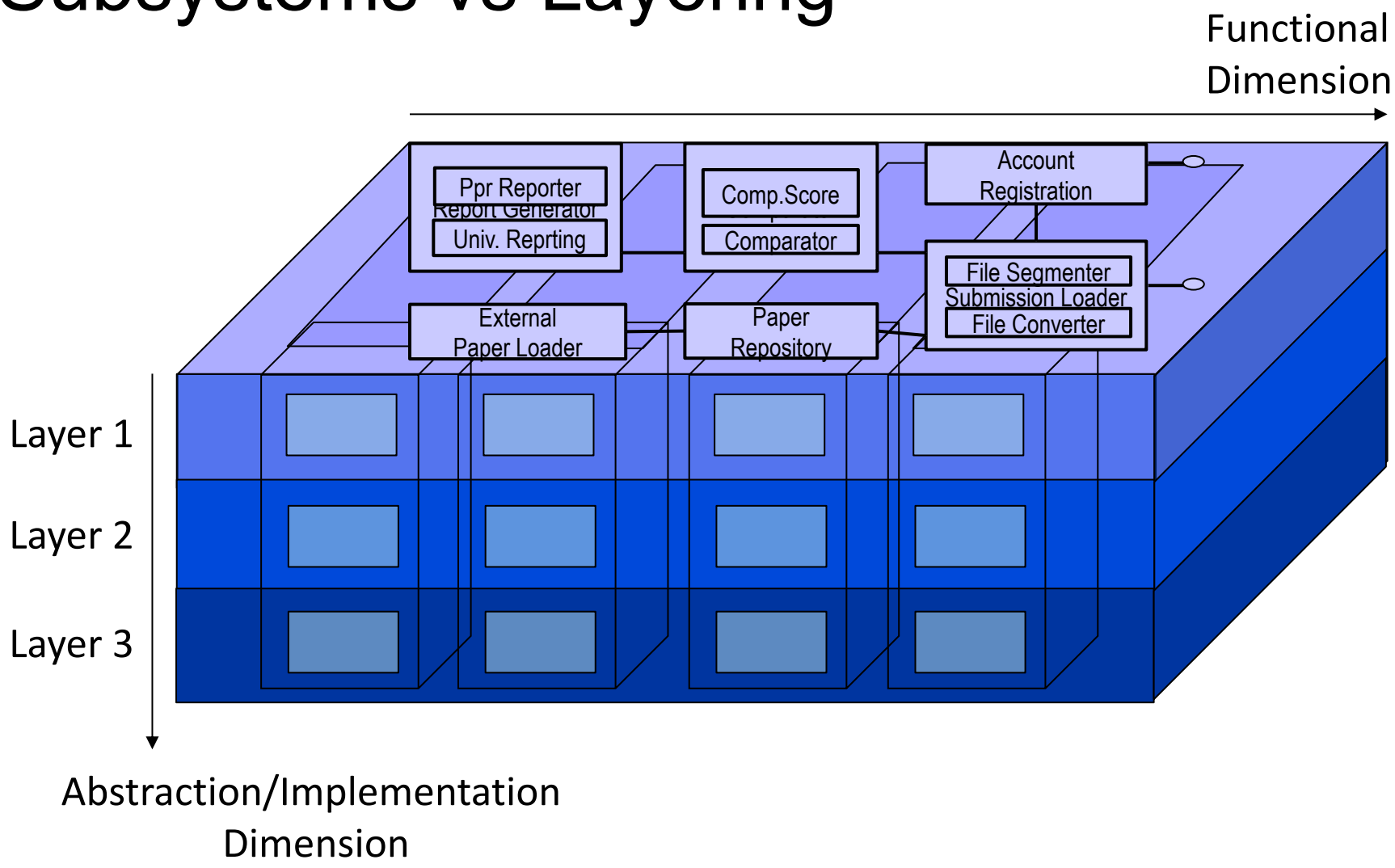


Where do these sub-subsystems go?




Report
Paper Reporter
University Reporting
Comparator
Compute Score
Comparator
Account Registration
Submission Loader
File Segmenter
File Converter
External Paper Loader
Paper Repository

Subsystems vs Layering



Analysis & Design

In this course

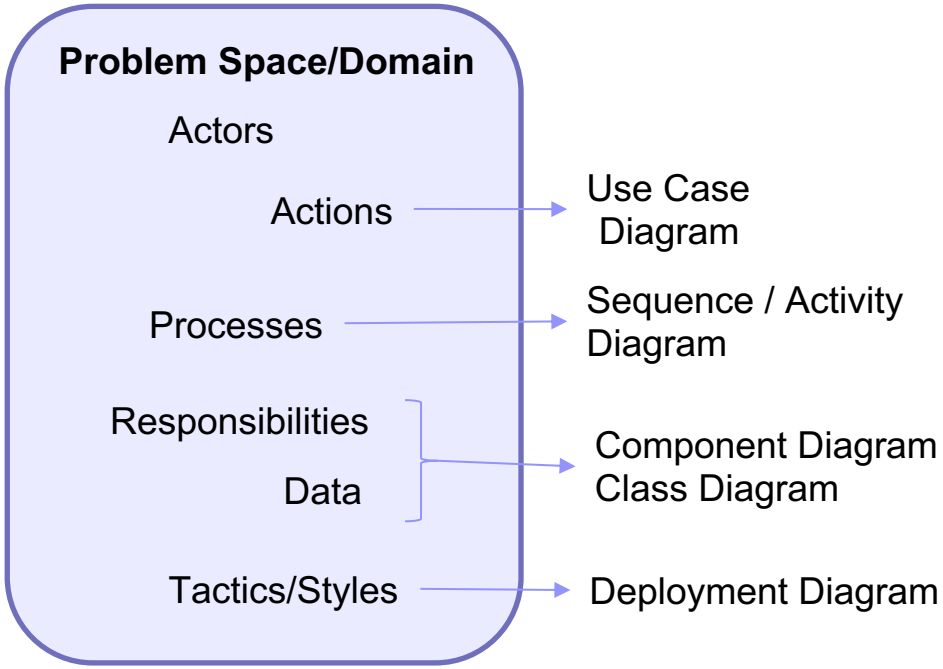
Analysis is for: understanding & describing the domain

describes ***what*** : main concept & their relations

Design is for: synthesis of executable solution

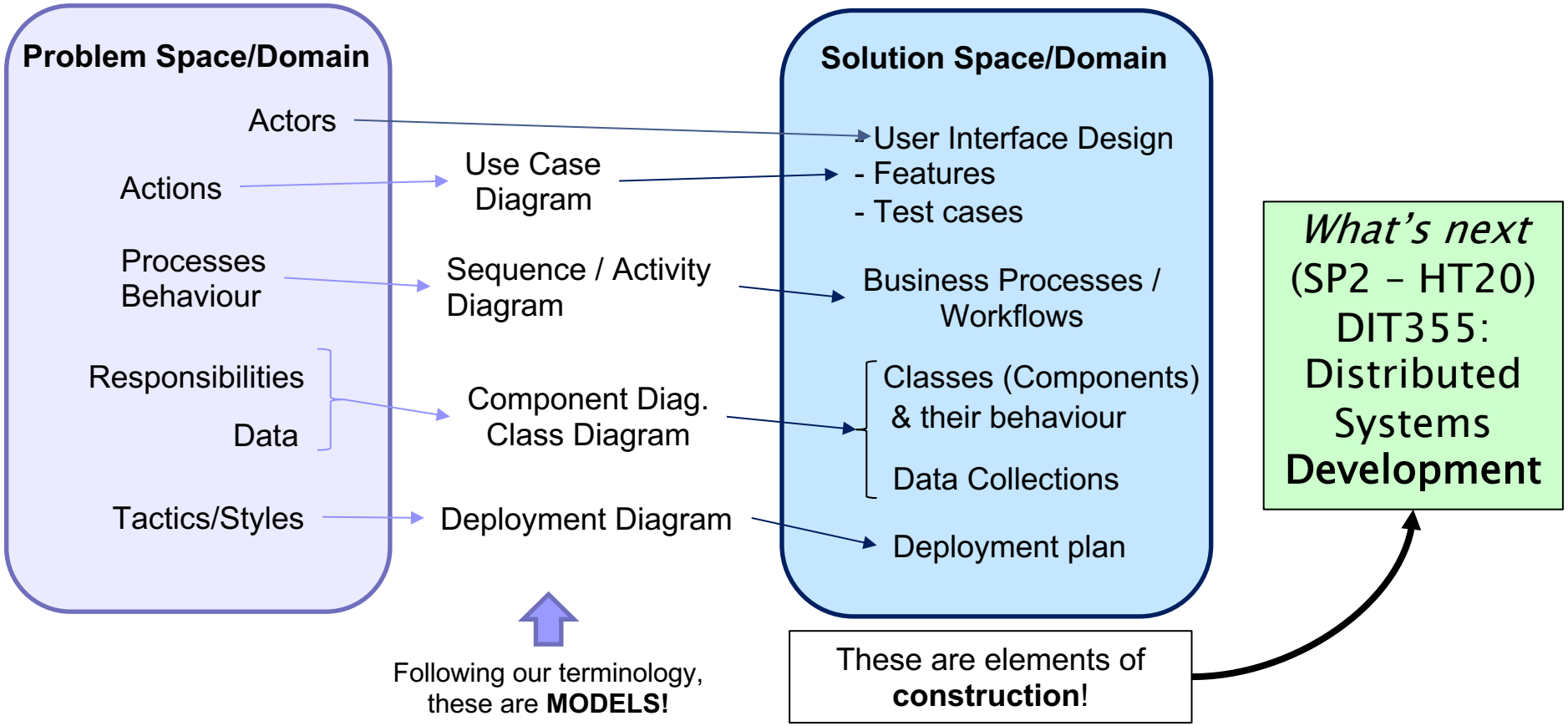
describes ***how*** a construction of a solution should work

From Analysis to Design



These are elements of **analysis:**
What is there?

From Analysis to Design



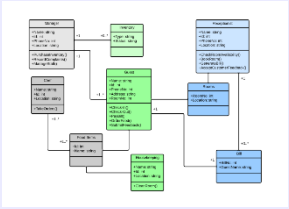
From Analysis to Initial Design

The OO paradigm has been designed such that Analysis and Design models look ‘alike’:
Classes that appear in a domain model, can also appear in a design model

Descriptive Models

Problem Space/Domain

Component/Class Diagram

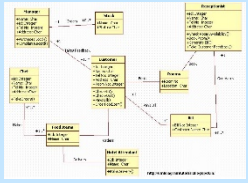


These are elements of **analysis:**
What is there?

Prescriptive Models

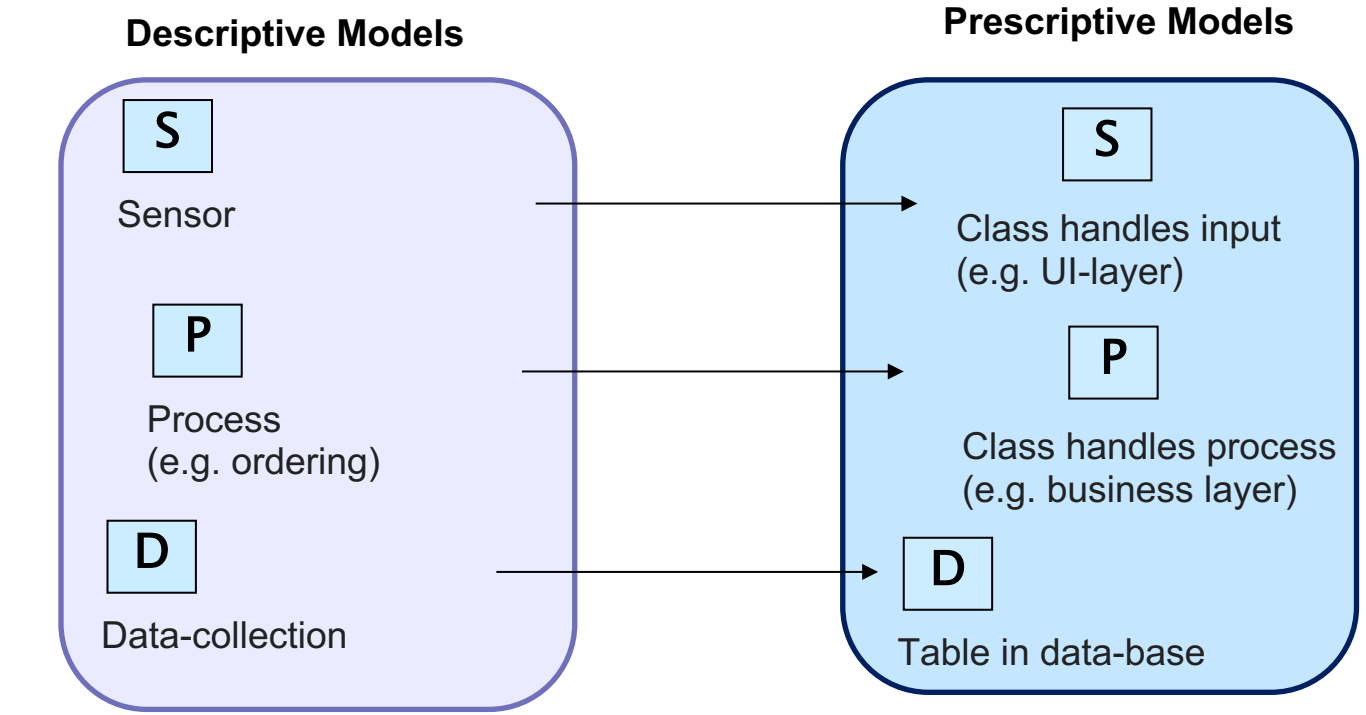
Solution Space/Domain

Components/Classes & their behaviour



These are elements of **construction**:
How will it work?

From Analysis to Design



These are elements of **analysis:**
What is there?

These are elements of **construction**:
How will it work?

Conceptual Integrity

Anywhere you look in your system, you can tell that the design is part of the same overall design style, theme, mood …is about **Design and Style Consistency in all dimensions of the system**

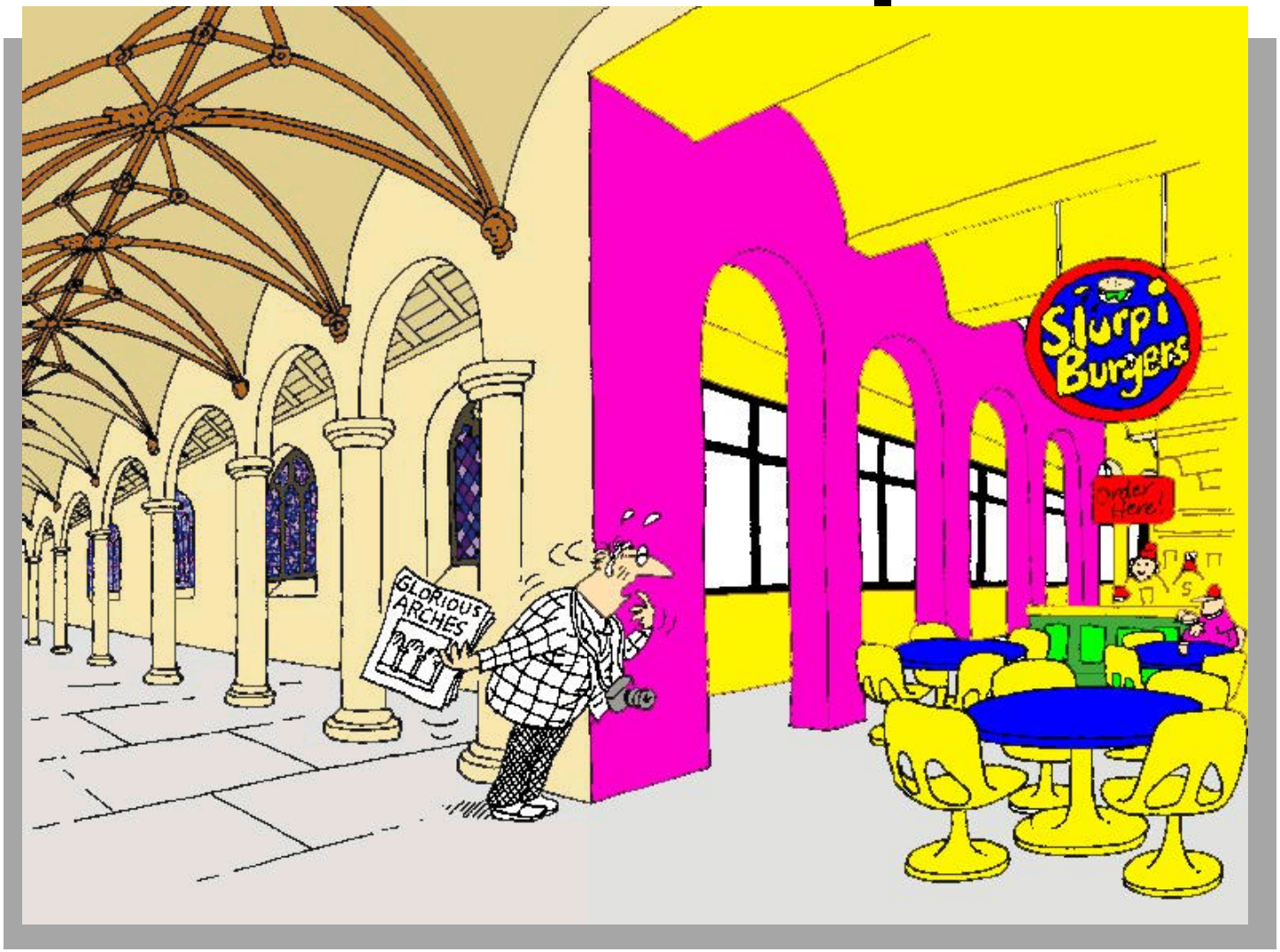
User interface, technologies, coding styles, naming conventions, directory structures, classes, components, interfaces, internal and external behavior, deployment…

Fed Brooks:
"It is better to have a system…reflect one set of design ideas, than to have one that contains many good but independent and uncoordinated ideas"

The Mythical Man-Month

Conceptual Integrity tries to limit the system complexity
Conceptual Integrity simplifies collaboration when creating software

Conceptual integrity counter example!

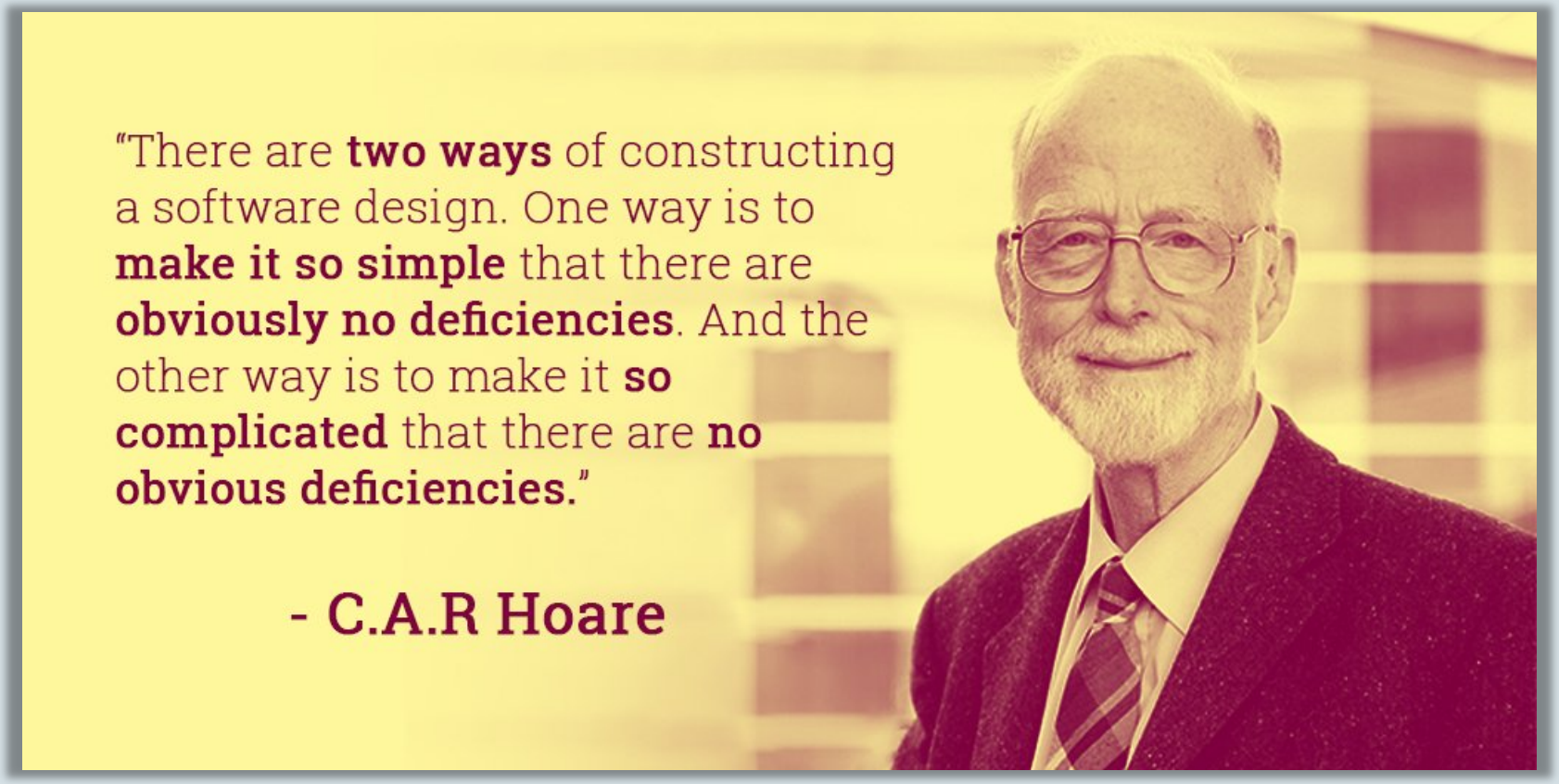


Conceptual Integrity in Software

Harmony

- Uniformity – where possible
  - Same solution-approach/tactic/design-principle applied to similar problems
  - For example in
    - patterns of communication & control/data-flow
    - Naming of methods, parameters, variables
    - Structure of API's
    - Layering
    - Exception-handling
    - User interface (dialogs)

Uniformity


"There are **two ways** of constructing a software design. One way is to **make it so simple** that there are **obviously no deficiencies**. And the other way is to make it **so complicated** that there are **no obvious deficiencies.**"
- C.A.R Hoare

CONTENTS

CONTENTS

What is an Architectural Style?

- Architectural styles are **collections of design conventions** for a set of design dimensions

  Some architectural styles emphasize different aspects such as:

  - Subdivision of **functionality**,
  - **topology**
  - **interaction/coordination** pattern between components

- An architecture can **use several** architectural styles
- Architectural styles are **not disjoint**
- Styles are **open-ended**; new styles may emerge

Architectural style

An *architectural style* is defined by:

a set of rules and constraints that prescribe

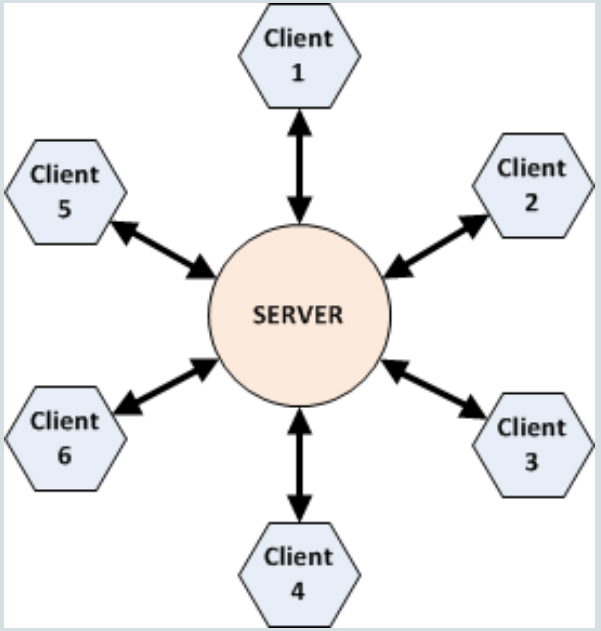
- **vocabulary/metaphor**: which types of components, interfaces & connectors must/may be used in a system.
  Possibly introducing domain-specific types
- **structure**: how components and connectors may be combined
- **behaviour**: how the system behaves
- **guidelines**: these support the application of the style (how to achieve certain system properties)

What is a Style?

Coordinated, aligned

CONTENTS

Client-Server Architectures



Nice source:

IT Architectures and Middleware:
Strategies for building Large Integrated Systems,
Chris Britton and Peter Bye, Addison Wesley, 2004

What is Client / Server?

Client: an application that makes requests (to the servers) and handles input/output with the system environment

Server: an application that services requests from clients

Client/Server System:
an application that is built from clients and servers

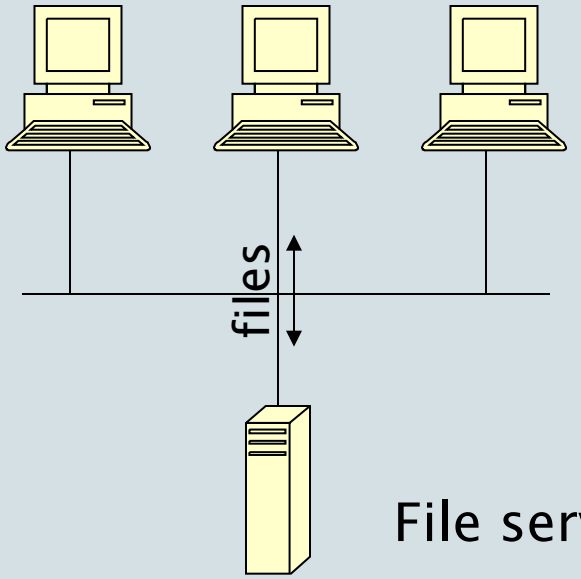
Typical application area:
distributed multi-user (business) information systems

In real-life: client *instructs/commands* the server.
Pitfall: keep an eye on (hidden) dependencies

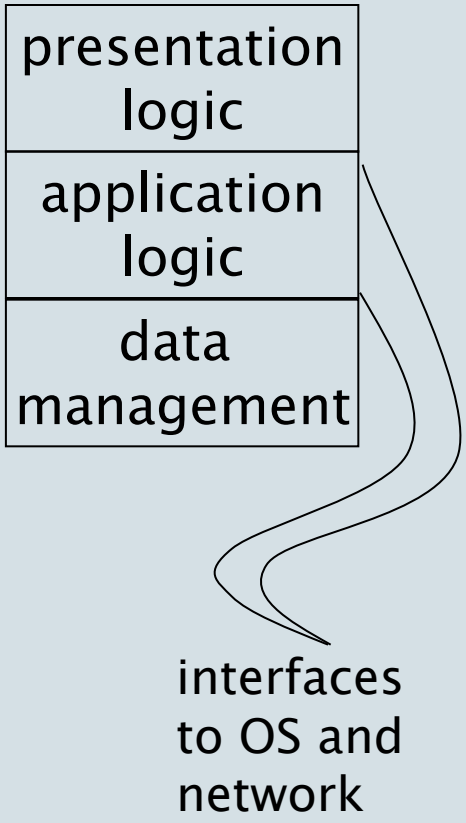
Why Client / Server?

±1980's

- multiple users want to share and exchange data
- first: attempt: shared file-server
- problem:
  scalability to ±10 due to contention for files and volume of data-transfer
- solution:
  perform processing on (file) server



3-tier Reference Model



presentation logic ([G]UI):
anything that involves system/user interaction
e.g. dialogs (management), forms, reports

application logic (data processing):
where the functionality of the application resides / where the actual computation of the system takes place

data management:
storing, retrieving and updating data

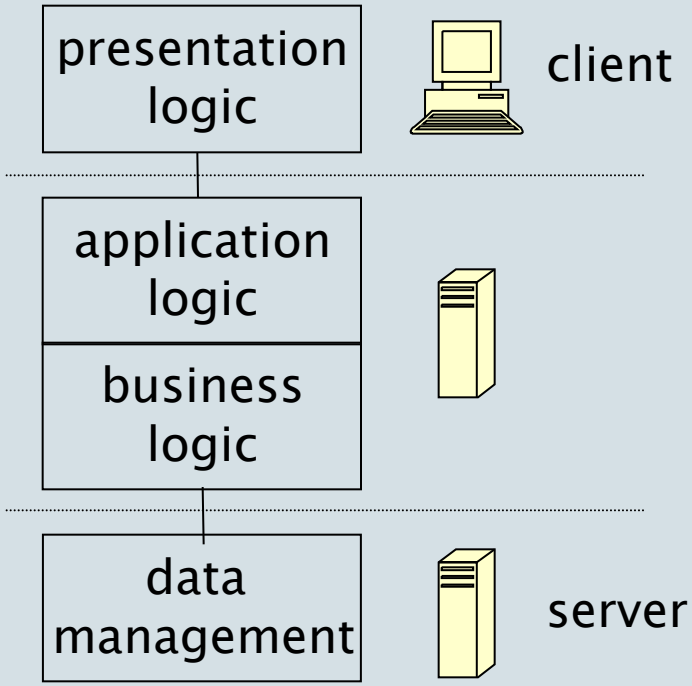
Client / Server Style

Concept: Separation of application in units of change

Components: presentation,
application logic,
business logic,
data management

Connector: 'uses' lower layer

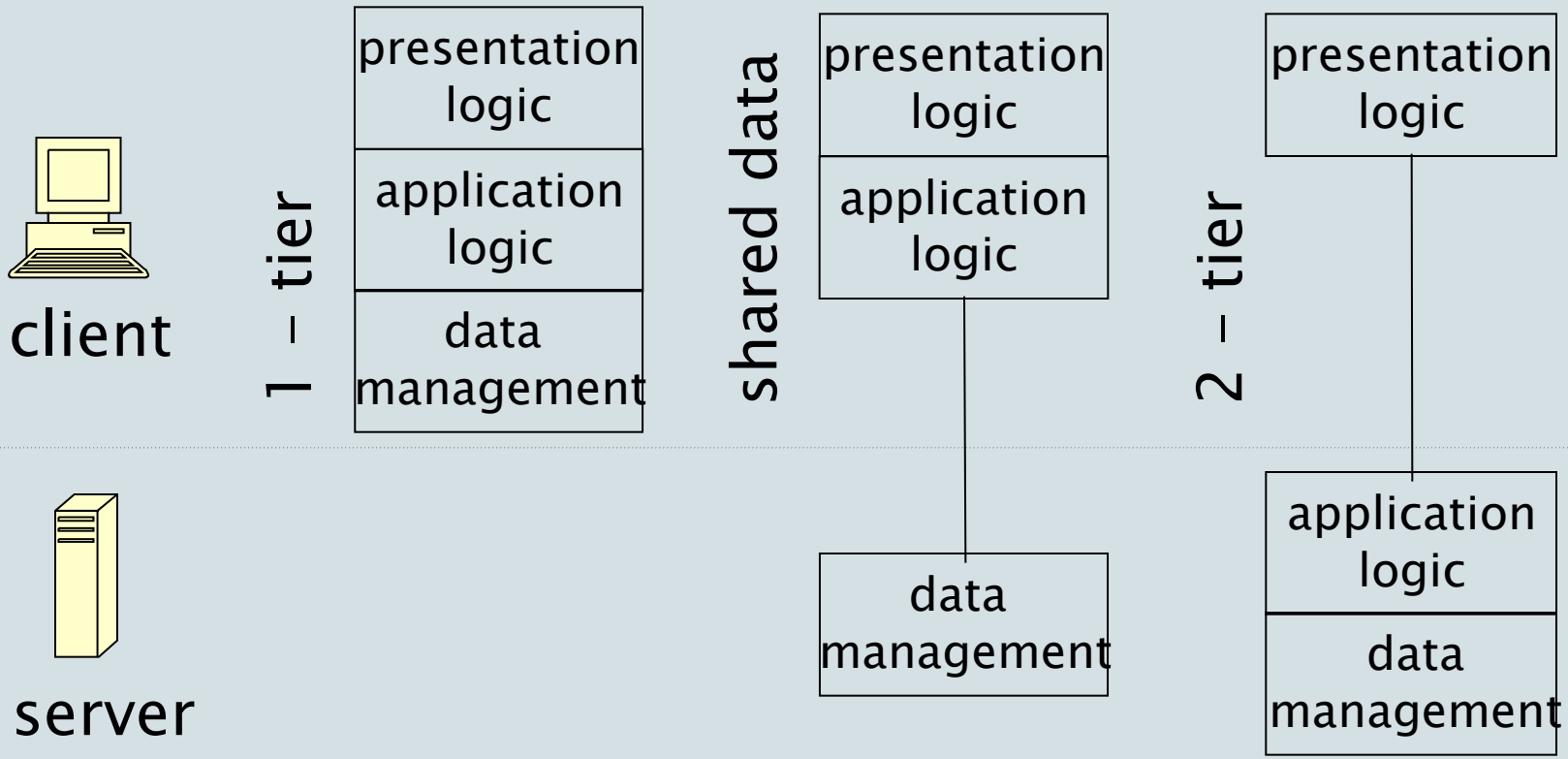
Interaction style: request/response



Deployment of C/S Model

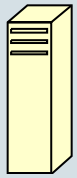
Rather than having the client do the processing …

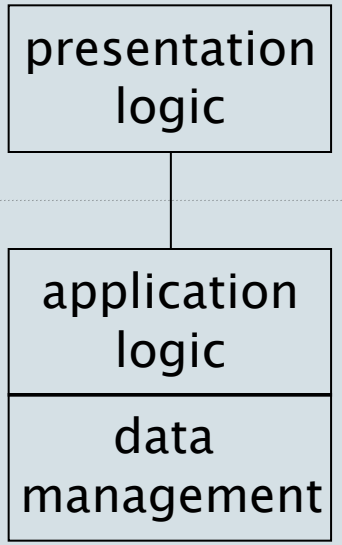
Move processing power to server such that the server sends a (condensed) response to request rather than a whole file



C/S Example: Thin Client

Thin Client C/S:
largest part of processing at the server-side





WWW Browser

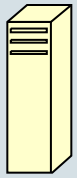
functional processing

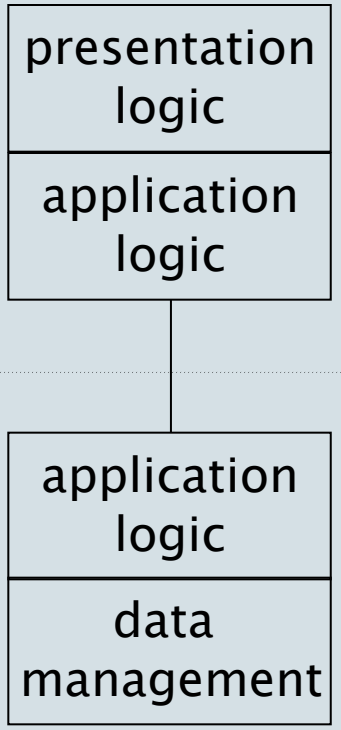
database storage and access

Network load: low
Config. Mngmnt: simple (only server)
Security: concentrated at server
Robustness: stateless clients => easy fault recovery

C/S Example: Thick Client

Thick Client:
significant processing
at the client-side





WWW Browser

application (specific)

application (generic)

database

Network load: high
Config. Mngmnt: complex (both client & server)
Security: complex (both client & server)
Robustness: clients have state => complex fault recovery

Client/Server in terms of Characteristics

Dependencies: client depends on the server

Topology: one or more clients may be connected to a server there are no connections between clients

Multiplicity: 1-to-1, directed

Synchronicity: synchronous or asynchronous

Mobility: easily supports client mobiliy

Binding: from compile to invocation time

Initiative: request (by client) / response (by server)

Periodicity: typically a-periodic (periodic possible)

Security: typically controlled at server, also possible at application/business layer

Example 3-tier System

Presentation tier

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier (application)

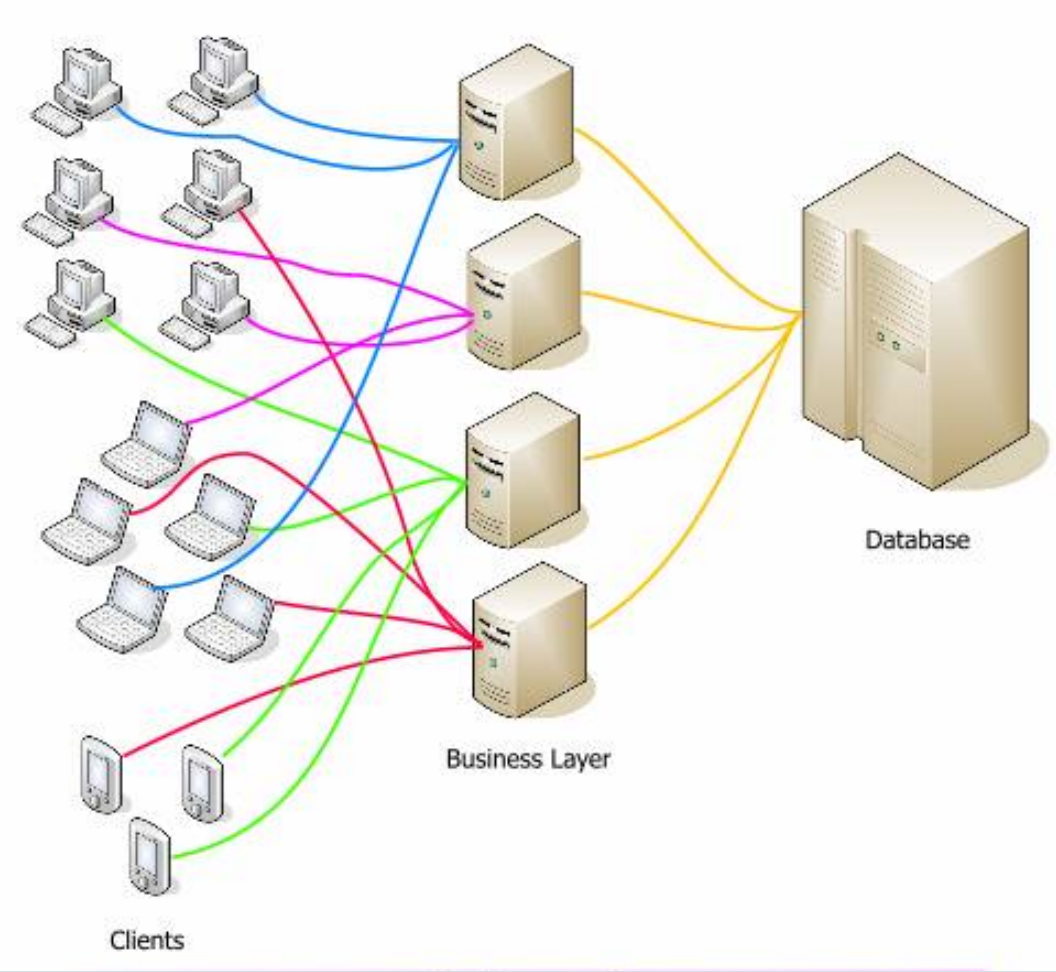
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

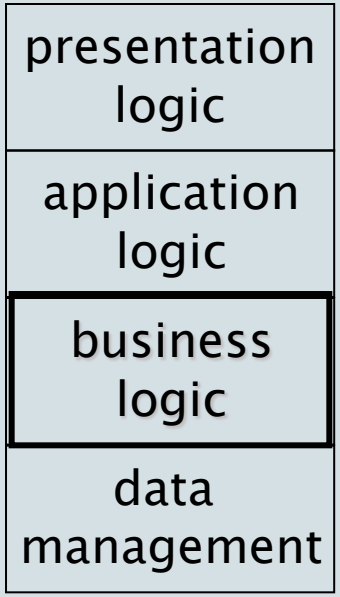
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

Diagram from Wikipedia, 2007

Deployment: Many physical clients and servers



Business Logic Layer

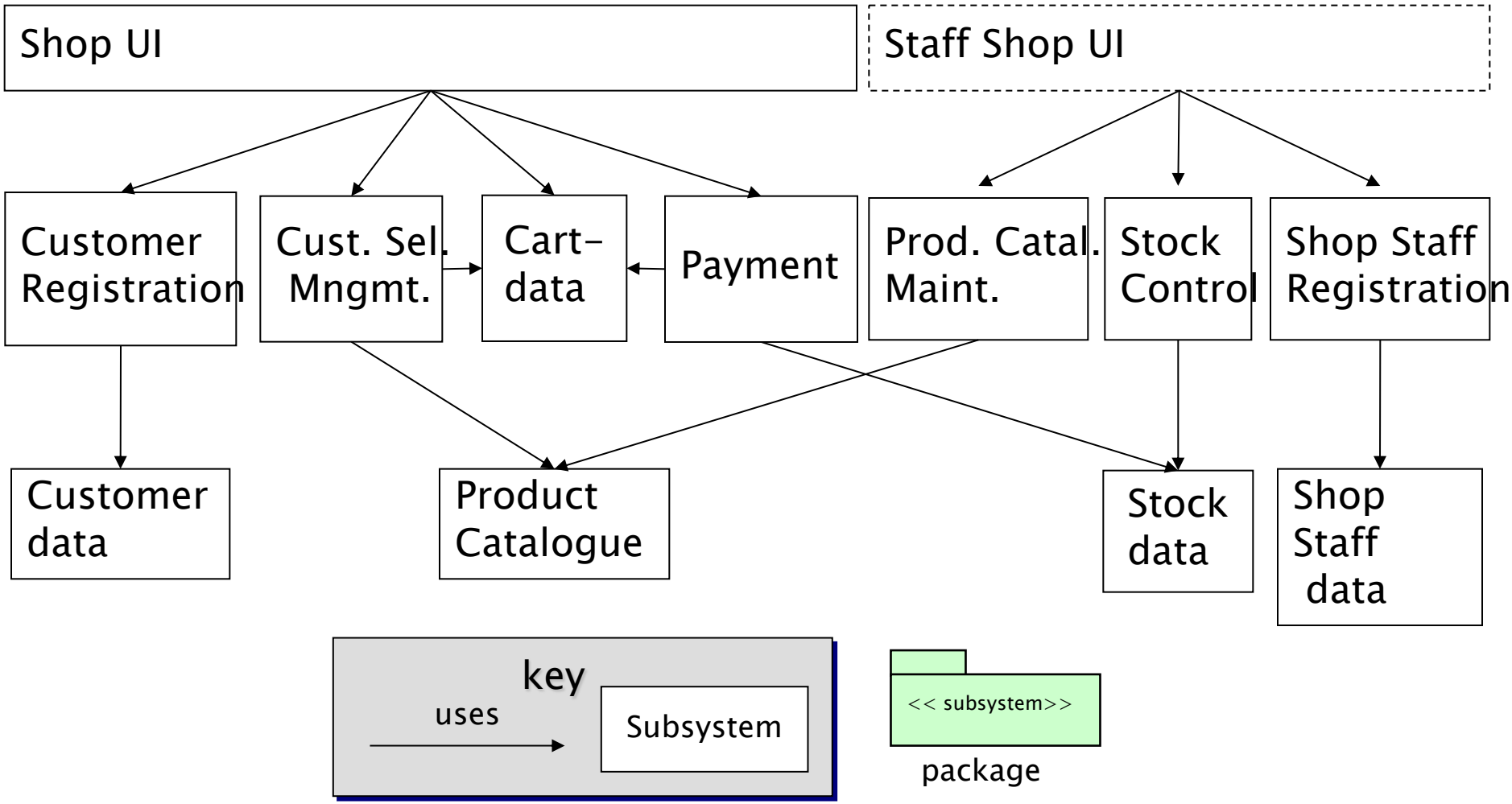


An even broader organizational scope of sharing and exchanging data requires
coordination across *multiple applications* and *databases*

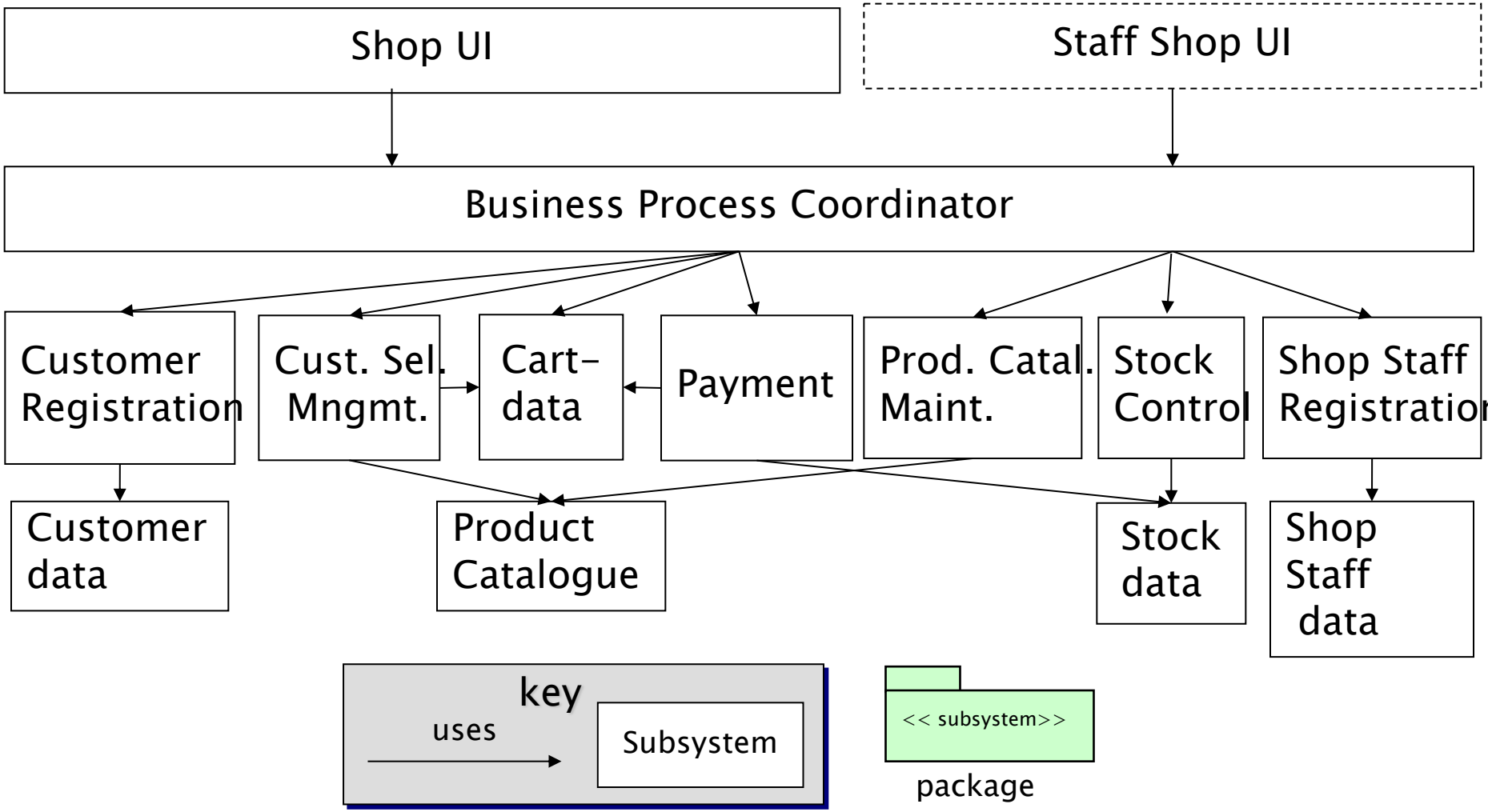
The complexity of the middle tier ranges from
reactive with little intelligence
(e.g. resource management and interconnection services)
to active with much intelligence
(active enforcement of global constraints and coordination of activities across applications e.g. workflow)

Advantage of business logic-tier:
changes to business are localized
(compared to intertwining with application logic)

Identification of Dependencies



Introduce layer for business logic



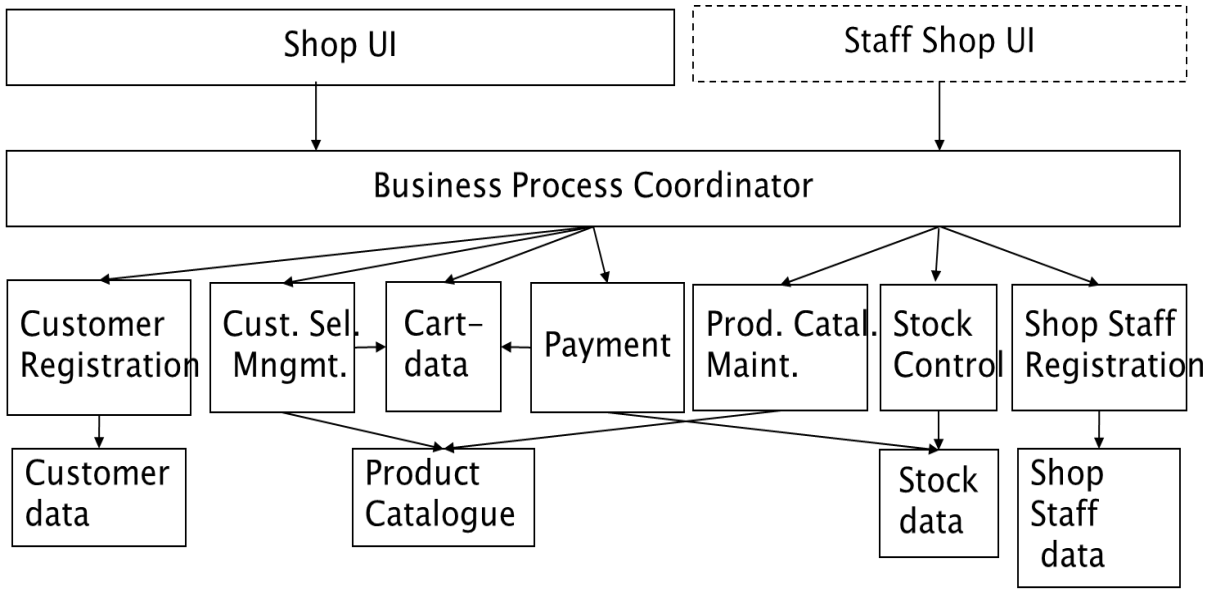
Stereotypes in layered CS-design

Interfacers

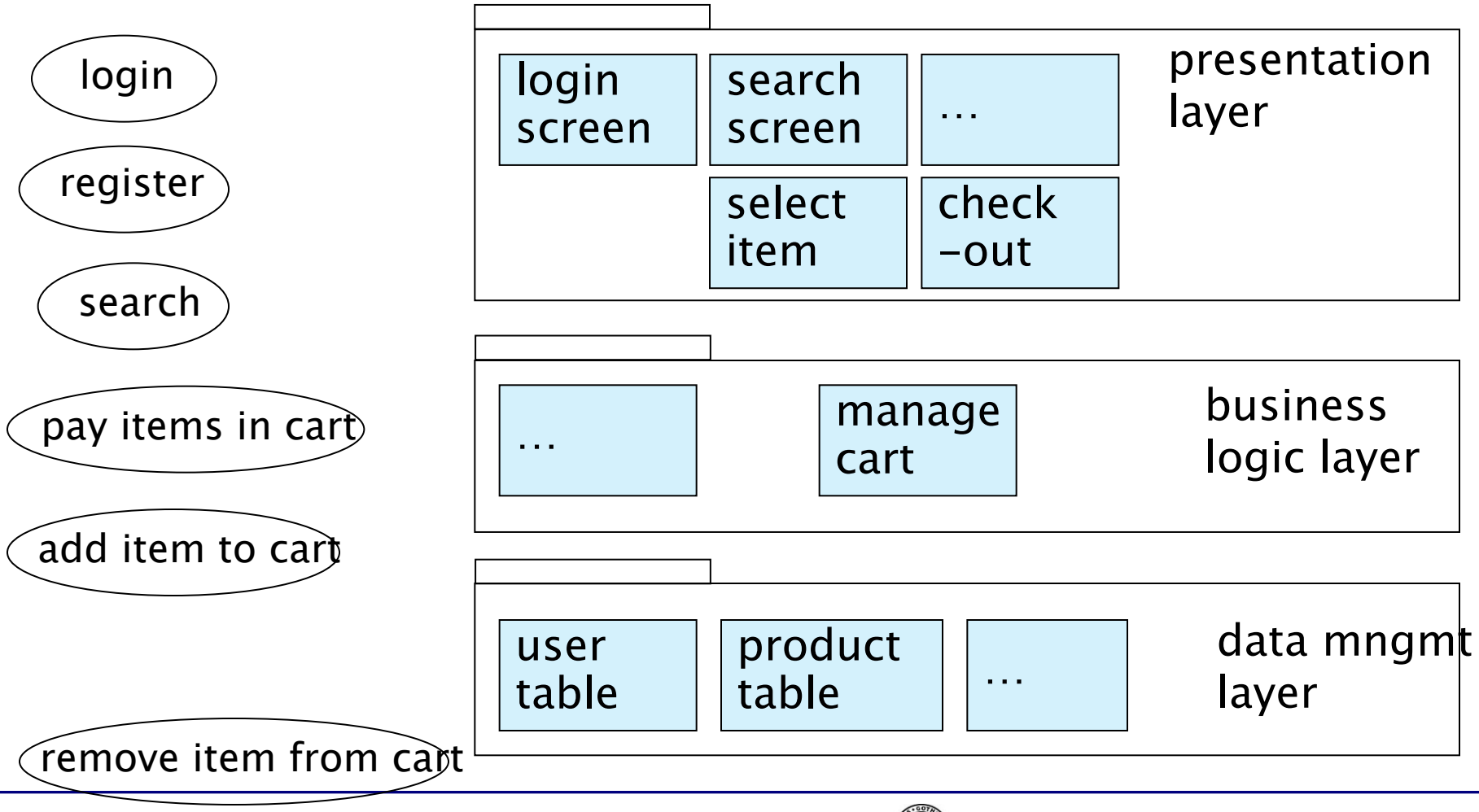
Coordinator

Service Providers

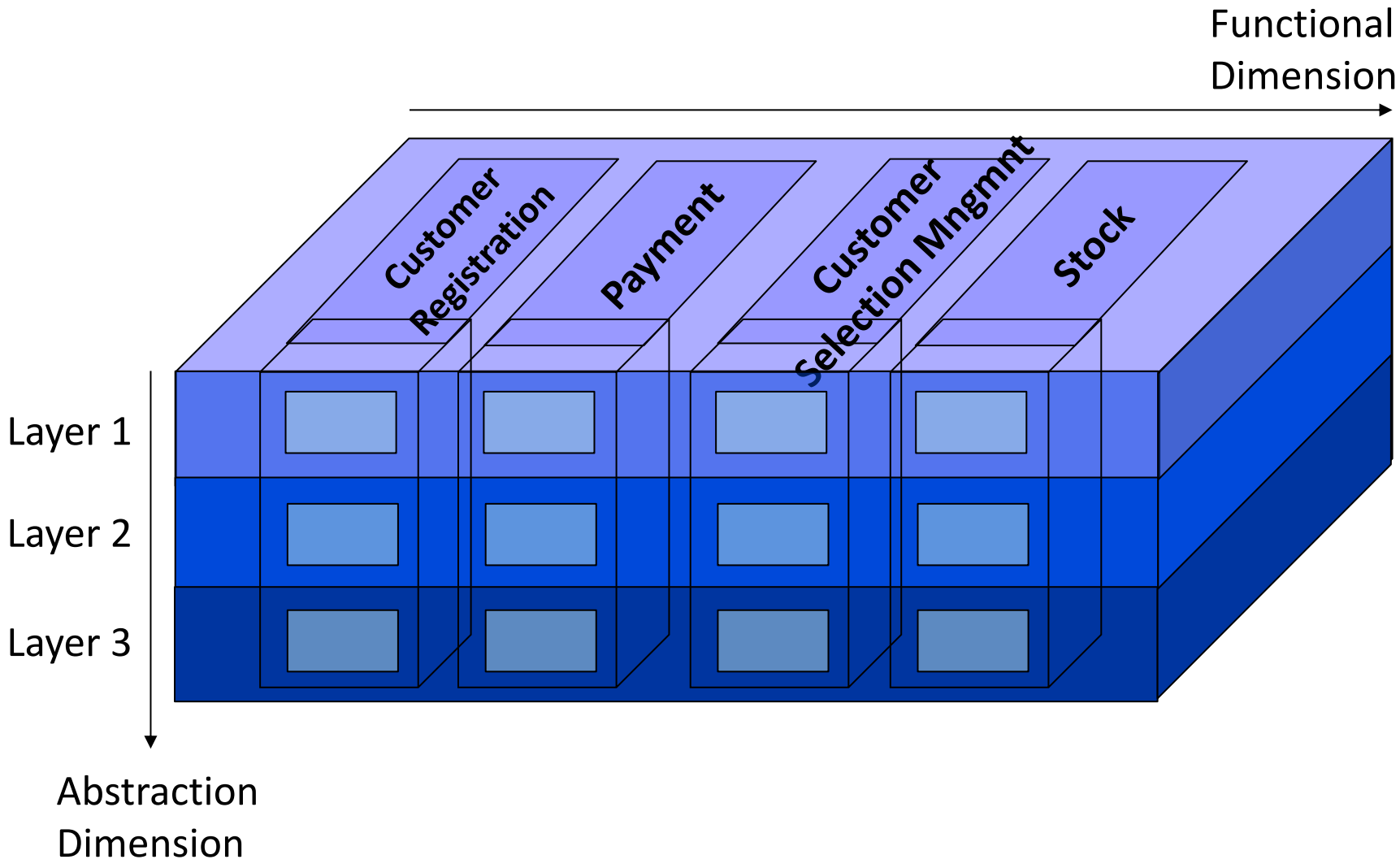
Information holders



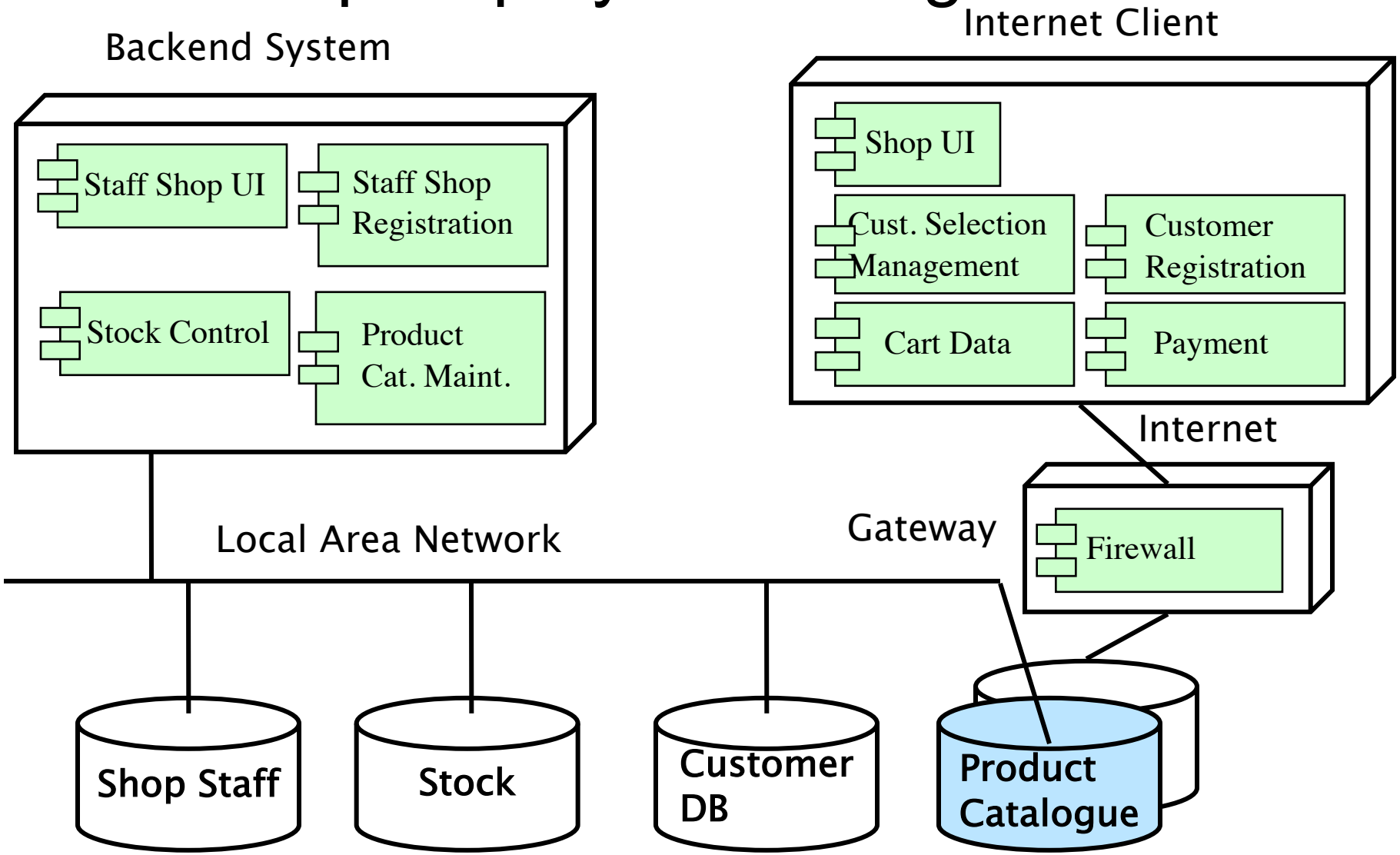
Identify support for Use Cases at different layers in the architecture



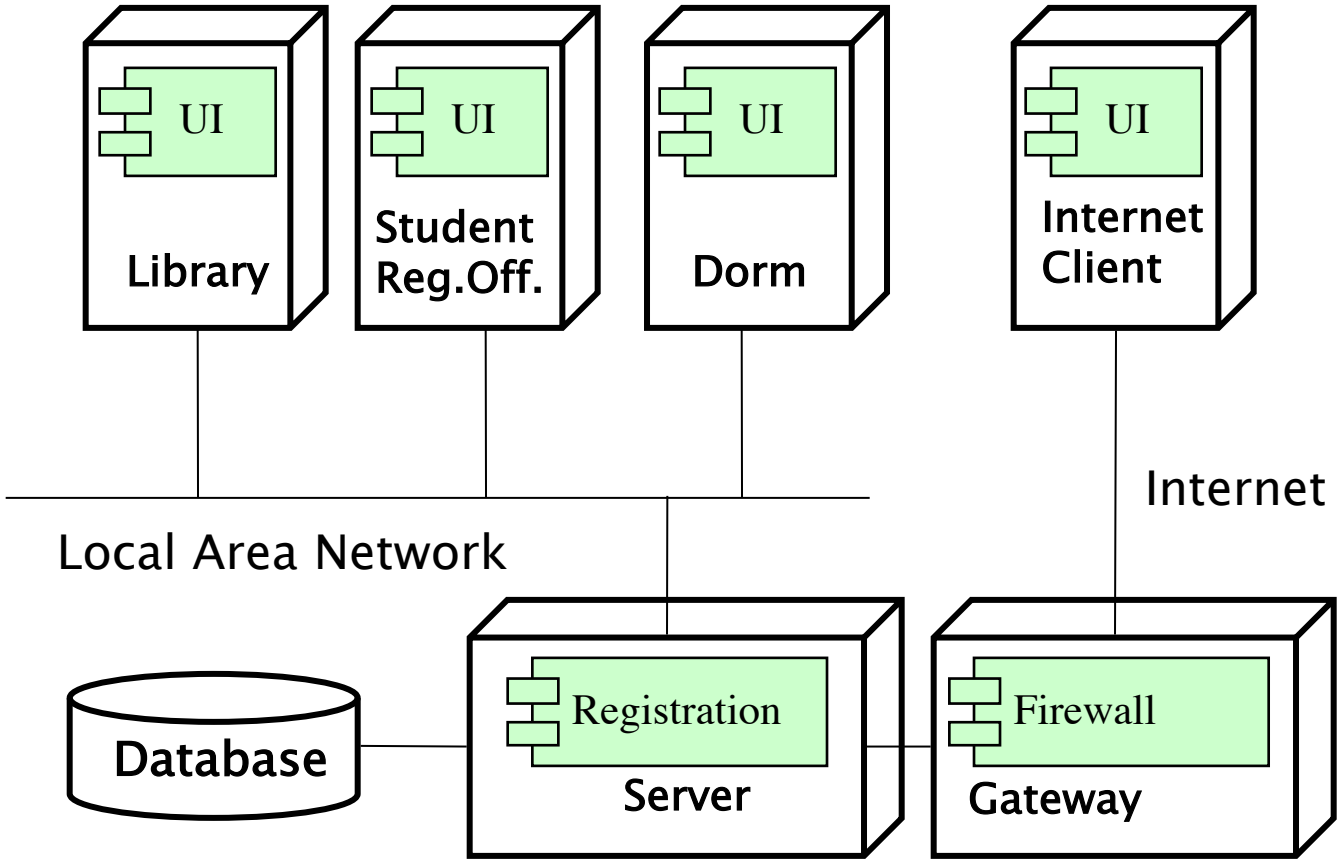
In which dimension(s) does C/S-style apply?



Web Shop Deployment Diagram



Deployment Example



Client / Server Summary

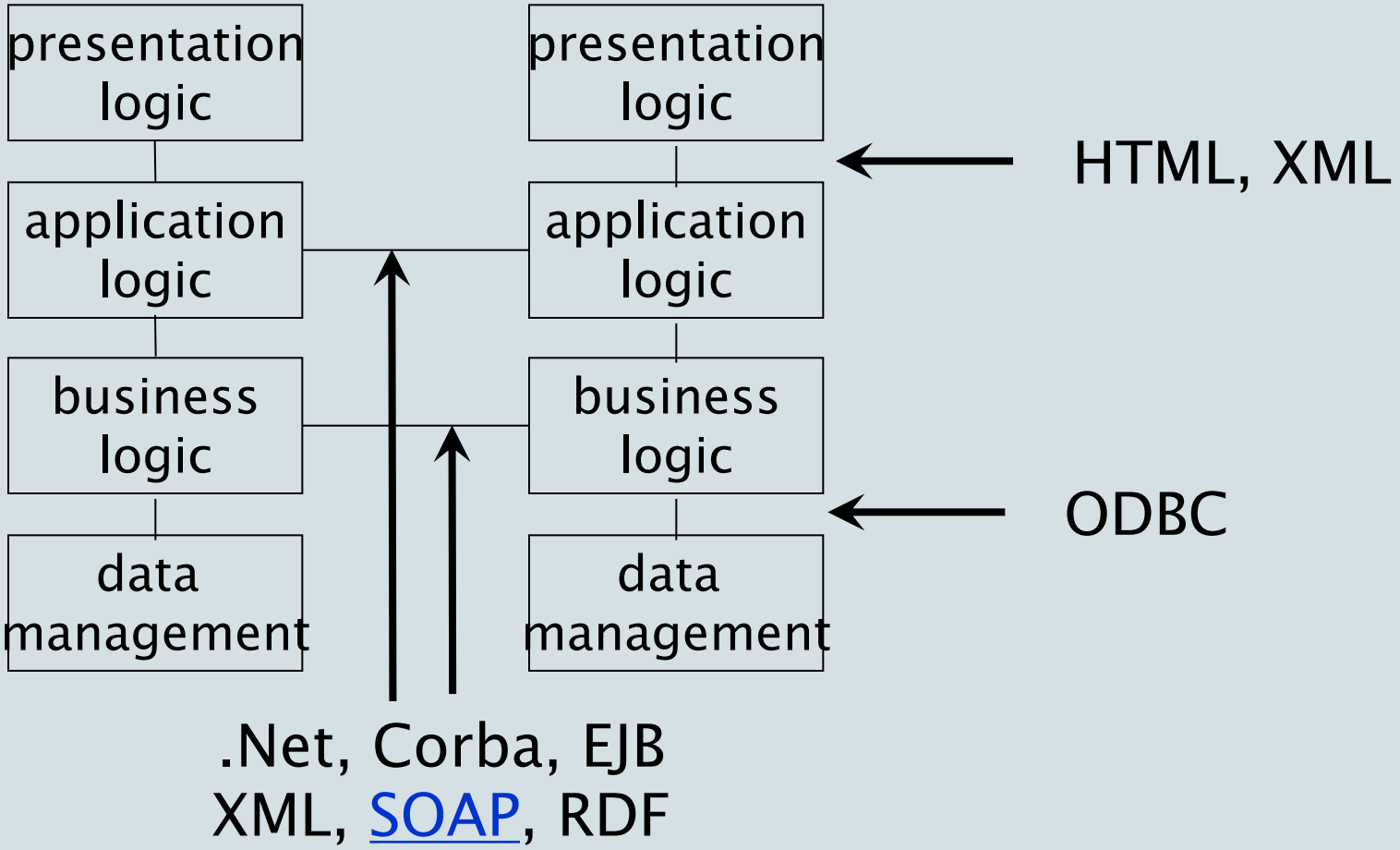
Task are mapped on platform where they are most efficiently handled

- presentation layer on client
- data management and storage on a server
- possible intermediate platforms for transaction multiplexing and global coordination

With the aim of obtaining

- scalability: changing number of clients
- interoperability: client may use data from multiple sources

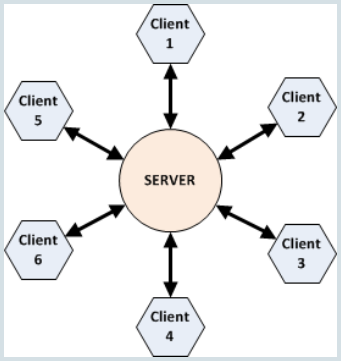
Interface Technologies



n-tier web applications

Layer	Functionality	Role	Technique
Client	User Interface	**Interfacer**	HTML, JavaScript
Presentation	Page Layout	**Interfacer**	JSP
Control	Command	**Coordinator**	Servlet
Business Logic	Business Delegate	**Controller**	POJO, Session EJB
Data Access	Domain Model	**Information Holder, Structurer**	JavaBean, Entity EJB
Resources	Database, Enterprise Services	**Service Provider**	RDBMS, Queues, Enterprise Service Bus

Client-Server



Advantages:

- centralized data access
  - higher security
- ease of maintenance
- scalability (in clients)
- interoperability (use of multiple sources)

Limitations:

- single point of failure

WWW Sources for C/S

C/S-FAQ:
http://www.faqs.org/faqs/client-server-faq/

C/S info @ Software engineering institute Carnegie Mellon Univ.
http://www.sei.cmu.edu/str/descriptions/clientserver_body.html

Summary

Decomposition
- Functional Decomposition vs Implementation Decomposition

Conceptual Integrity
- uniformity, harmony, consistency in design

Architectural Styles
- Every Architect should have a standard set of architectural styles in his/her repertoire
  - Client/Server

The choice for a style can make a big difference in the *quality properties* of a system