Systems Development

Lecture 8: Architectural Design, Criteria and Components

Contents

- Summary of last lecture
- Why are we making the descriptions?
- Architectural design
- The Criteria activity
- The Components activity

Contents

- Summary of last lecture
  - Difficulties in exercises
  - The Function activity
  - Event – use case – function
- Why are we making the descriptions?
- Architectural design
- The Criteria activity
- The Components activity

Difficulties in Exercises

- Final state for an object: general and in analysis and design
- Iteration in statechart diagrams: how should it be modelled
- Item-Descriptor pattern and the behaviour of the classes
- Actor: what is and what is not

Functions: Results

- Primary: a complete list of functions

Planning		
Make schedule	Very complex	Update
Calculate schedule consequences	Complex	Signal
Find working hours from previous period	Medium	Read
Enter contents into schedule	Complex	Update
Erase schedule	Simple	Update
Query earlier schedules	Medium	Read
Make appointment	Medium	Update
Cancellation	Simple	Update
Query possible appointments	Complex	Read
Register treatment	Simple	Update
Create customer	Simple	Update
Query customer information	Medium	Read
Employment	Simple	Update
Retirement	Simple	Update
Update apprentice information	Simple	Update

- Secondary: specifications of complex functions

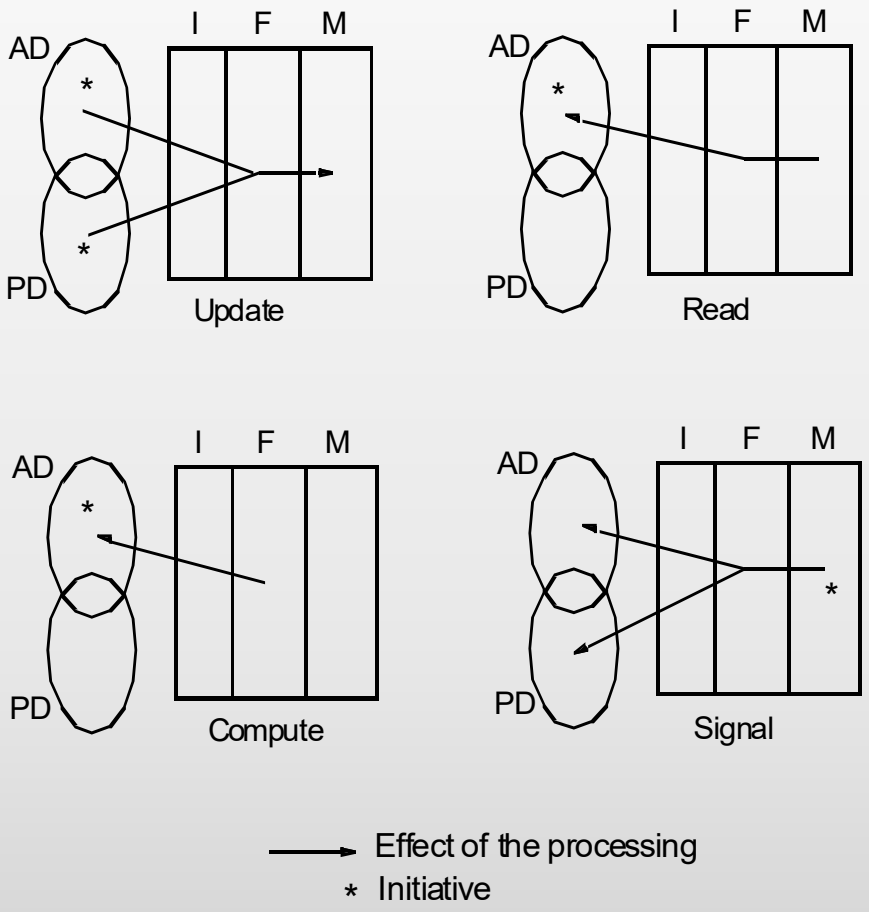
Query possible reservations:
given time or date or employee-name
search objects in time period-available and select those
who belong to employee-name, if known
have date, if known
cover point in time, if known
result objects of time period-available that fulfill the criteria

- Assessing complexity:
  - Simple: sets or reads the value of an attribute in an existing object
  - Medium: creates a new object and connects it by object structure(s) to other objects
  - Complex: reads or creates several objects

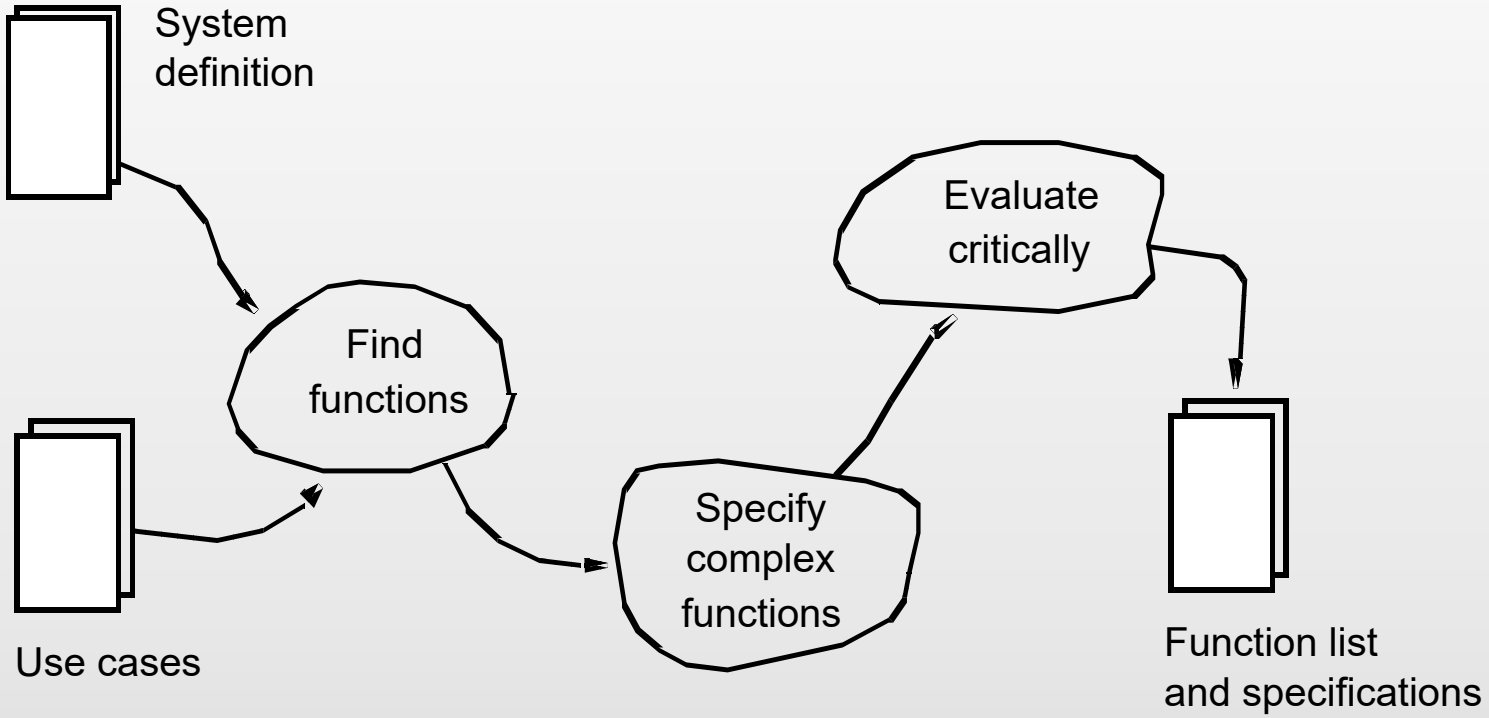
Key Concepts: Functions and Function Types

Function:
A facility for making a model useful for actors:

- A ressource for actors
- Uses the model component in order to support use cases



Functions: Activities



Evaluate Systematically

- Completeness:
  - Let the users review the list of functions
  - Use the questions for each function type to exhaust that category
  - Compare with the system definition, the model, and the use cases
- Make experiments and prototypes
  - To check the use cases
  - To check the set of function

Functions: Summary

Purpose	• To determine the system's information processing capabilities.
Concept	• Function: A facility for making a model useful for actors.
Principles	• Identify all functions • Specify only complex functions • Check consistency with uses cases and the model
Result	• A complete list of functions with specification of the complex functions.

Appreciate the Difference: Event – Use Case – Function

- A source of confusion
- All three describe dynamic aspects
- They are connected

but …

- They belong to separate domains
- They are all needed because they emphasize different parts of the requirements to the system
- but keep them separate

- Example: order processing system
  - Event
    Ordered – a customer has entered into a legally binding agreement at a point in time
  - Use case
    Enter order – a user in the application domain applies the system to make an order for a customer
  - Function
    Create order – in the system's model, an object of the class Order is created

Quiz 6 Overview

Quiz 6

Average

3.38 (of 4.00) of 56 finished attempts (of 159)

Best result (0.67-1.00)

1 (1.00) Which are function types?
2 (0.92) Match the function types with the diagrams:
3 (0.86) A read function is:

Middle result (0.34-0.66)

4 (0.59) Which pattern(s) are used in the diagram?

Worst result (0.00-0.33)

None

Quiz 3 and 4

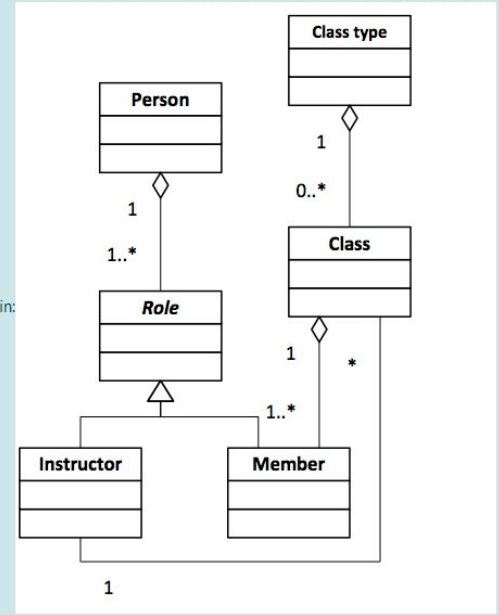
A read function is:

Select one:

- (●) a. activated by an actor in the application domain and displys relevant parts of the model
- ( ) b. activated by an event in the problem domain and displays relevant parts of the model
- ( ) c. activated by a signal function and displys information about the application domain
- ( ) d. activated in the model and displays a calculation

Clear my choice

A proposed system shall manage members, instructors, and classes in a gym. The developers have built this class diagram following their analysis of the problem domain:



Which pattern(s) are used in the diagram?

Select one or more:

- [ ] a. Material
- [ ] b. Composite
- [ ] c. Stepwise role
- [x] d. Item-descriptor
- [ ] e. Stepwise relation
- [x] f. Role
- [x] g. Relation
- [x] h. Hierarchy

Contents

- Summary of last lecture
- Why are we making the descriptions?
  - Use of descriptions (so far)
- Architectural design
- The Criteria activity
- The Components activity

Use of Descriptions (so far)

- System definition used in:
  - Evaluation of candidates for classes and events (Chapter 3)
  - Finding actors and use cases (Chapter 6)
  - Finding functions (Chapter 7)
- Event table used in:
  - Finding candidates for structures (Chapter 4)
  - Describing behavioural patterns (Chapter 5)
- Class diagram used in:
  - Describing behavioural patterns (Chapter 5)
  - Finding interface elements (Chapter 8)
- Behavioural patterns used in:
- Actors used in:
  - Describing use cases (Chapter 6)
- Use cases used in:
  - Finding functions (Chapter 7)
  - Finding interface elements (Chapter 8)
- Function list used in:
  - Finding interface elements (Chapter 8)

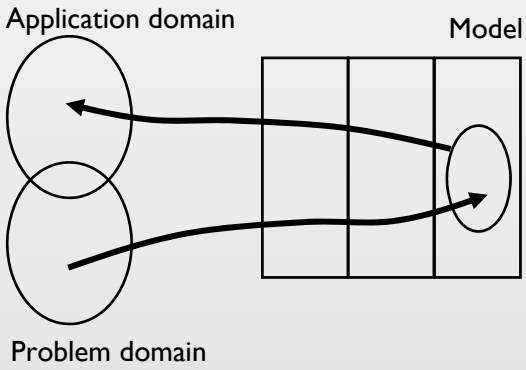
Contents

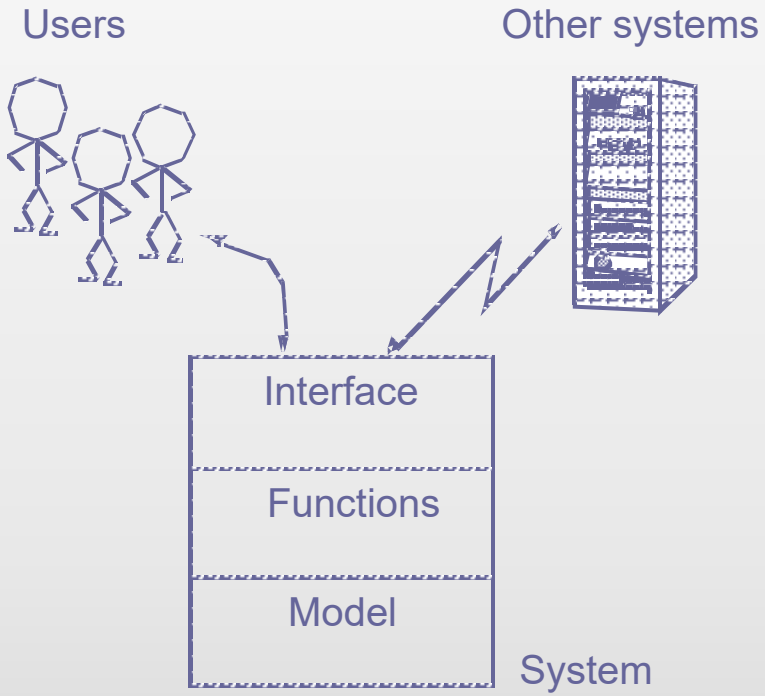
- Summary of last lecture
- Why are we making the descriptions?
- Architectural design
  - Key concepts
  - Views
  - Objects
  - Activities
- The Criteria activity
- The Components activity

Architectural Design: Key Concepts

System:

A collection of components that implements modeling requirements, functions, and interfaces.





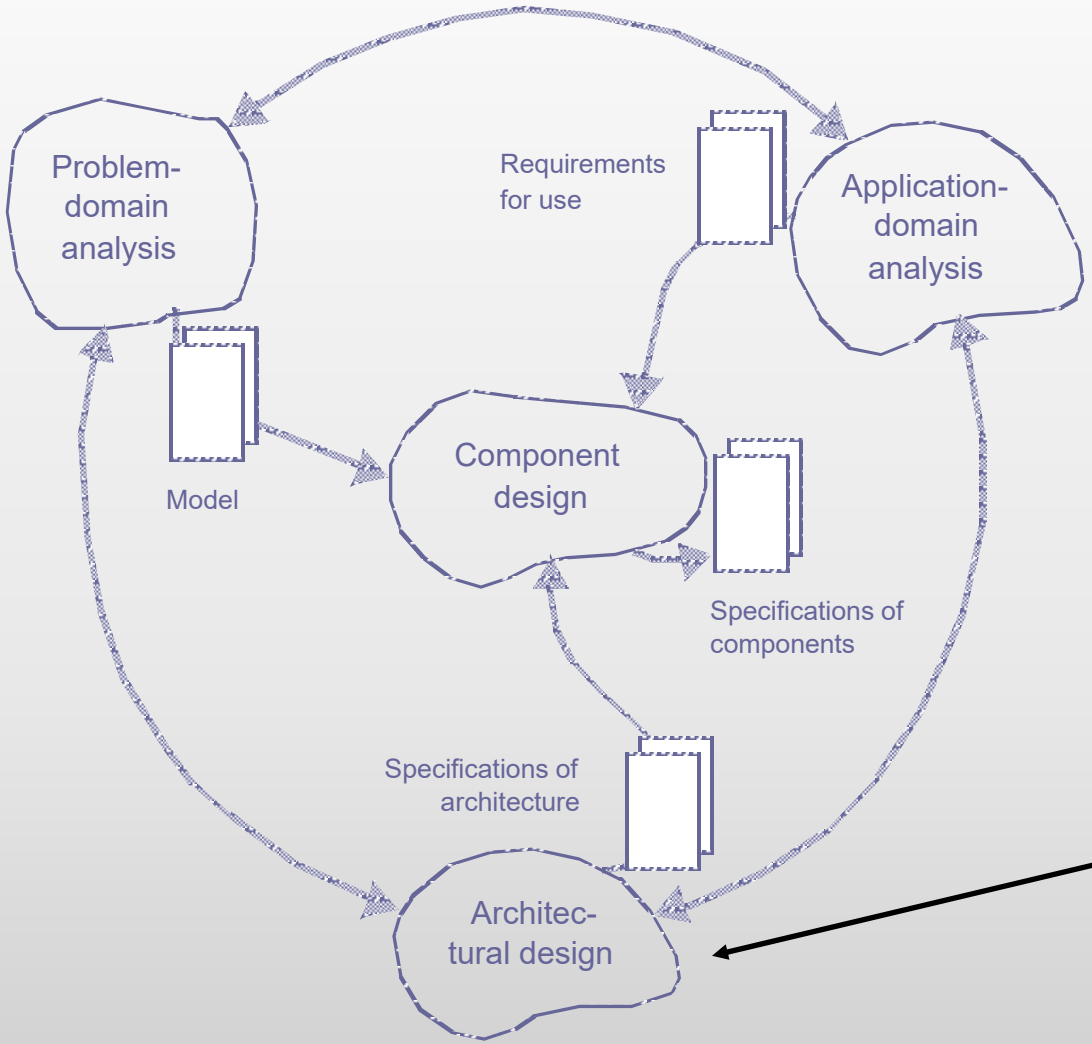
Architecture:

A general structure that is later developed further

Objects in Analysis and Design

- Analysis:
  - Phenomena outside the computer system
  - Identity: identifies an object
  - State: the qualities that characterise an object
  - Behavior: the events an object have performed or suffered
- Design (and programing):
  - Phenomena inside the computer system
  - Identity: gets access to an object
  - State: the values of the object's attributes and object structures
  - Behavior: the operations an object can perform on request and offers to other objects (methods)

Architectural Design: Activities



Architectural Design: Summary

Purpose	• To structure a computerized system.
Concepts	• Criterion: A preferred property of an architecture. • Component architecture: A system structure composed of interconnected components. • Process architecture: A system-execution structure composed of interdependent processes.
Principles	• Define and prioritize criteria. • Bridge criteria and technical platform. • Evaluate designs early.
Results	• Structures for a system's components and processes.

Contents

- Summary of last lecture
- Why are we making the descriptions?
- Architectural design
- The Criteria activity
  - Results
  - Key concepts
  - Activities
- The Components activity

Criteria: Result

- A collection of prioritized criteria

Criterion	*Very important*	*Important*	*Less important*	*Irrelevant*	*Easily fulfilled*
Usable	X				
Secure			X		
Efficient					X
Correct		X			
Reliable			X		
Maintainable			X		
Testable			X		
Flexible			X		
Comprehensible		X			
Reusable			X		
Portable	X				
Interoperable				X	

Criteria: Key Concepts

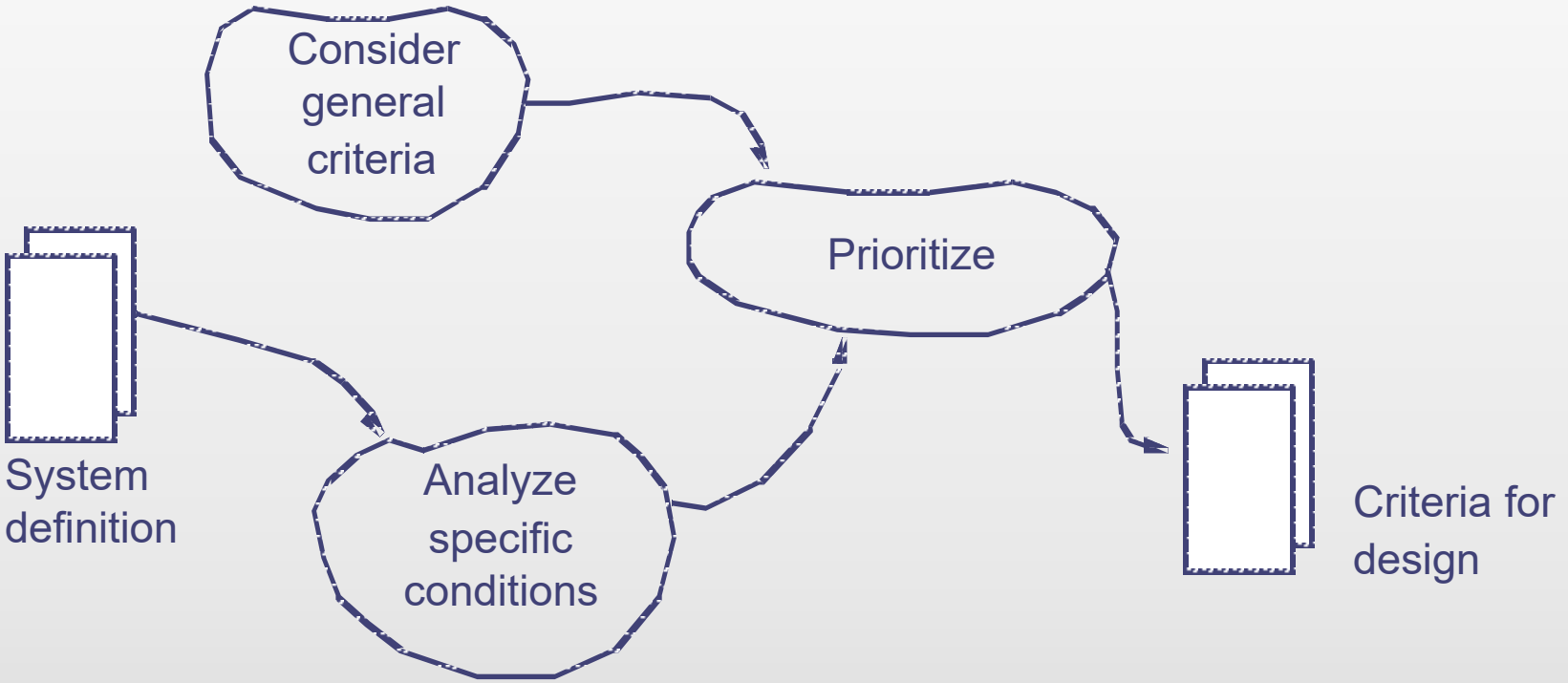
General criteria (McCall)

- Usable
- Secure
- Efficient
- Correct
- Reliable
- Maintainable
- Testable
- Flexible
- Comprehensible
- Reusable
- Portable
- Interoperable

Specific criteria in OOA&D

- Usability
  - The system as a whole
  - The users' needs
  - The technical platform
- Flexibility
  - Consequences of changes
  - Modular design
- Comprehensibility
  - Overview
  - Abstraction
  - Use of patterns

Criteria: Activities



Analyse Specific Conditions

Typical conditions for design of a system's architecture

Technical	• Existing hardware, basic software, and systems. • Reuse of patterns and existing components. • Use of purchased standard components.
Organizational	• Contractual arrangements. • Plans for continued development. • Division of work between developers.
Human	• Design competences. • Experience with similar systems. • Experience with technical platform.

Prioritize

Criterion	*Very important*	*Important*	*Less important*	*Irrelevant*	*Easily fulfilled*
Usable					
Secure					
Efficient					
Correct					
Reliable					
Maintainable					
Testable					
Flexible					
Comprehensible					
Reusable					
Portable					
Interoperable					

- Emphasize what is more and less important
- Not all criteria can be prioritized as important
- Why not?

Prioritize: Example

- Conference administration (Chapter 19)
- Usable: different volunteers; many over time
- Portable: used at each conference in different venues
- Correct: must fulfill the specification as the local users cannot identify and solve errors
- Comprehensible: must facilitate easy changes of the system
- Efficient: simple system for administrative tasks
- Interoperable: stand-alone system

Criterion	*Very important*	*Important*	*Less important*	*Irrelevant*	*Easily fulfilled*
Usable	X				
Secure			X		
Efficient					X
Correct		X			
Reliable			X		
Maintainable			X		
Testable			X		
Flexible			X		
Comprehensible		X			
Reusable			X		
Portable	X				
Interoperable				X	

- The rest: less important because of the nature of the problem and application domains + the stable nature of these domains

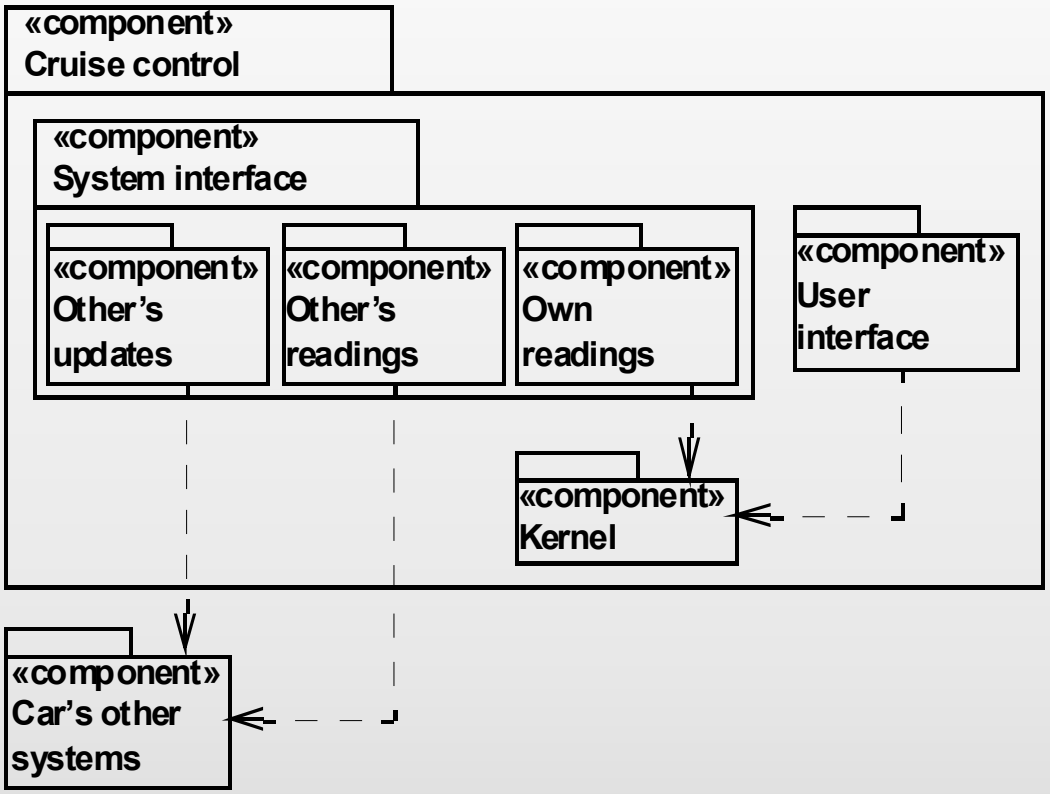
Criteria: Summary

Purpose	• To set design priorities.
Concepts	• Criterion: A preferred property of an architecture. • Conditions: The technical, organizational, and human opportunities and limits involved in performing a task.
Principles	• A good design has no major weaknesses. • A good design balances several criteria. • A good design is usable, flexible, and comprehensible.
Results	• A collection of prioritized criteria.

Contents

- Summary of last lecture
- Why are we making the descriptions?
- Architectural design
- The Criteria activity
- The Components activity
  - Results
  - Key concepts
  - Activities

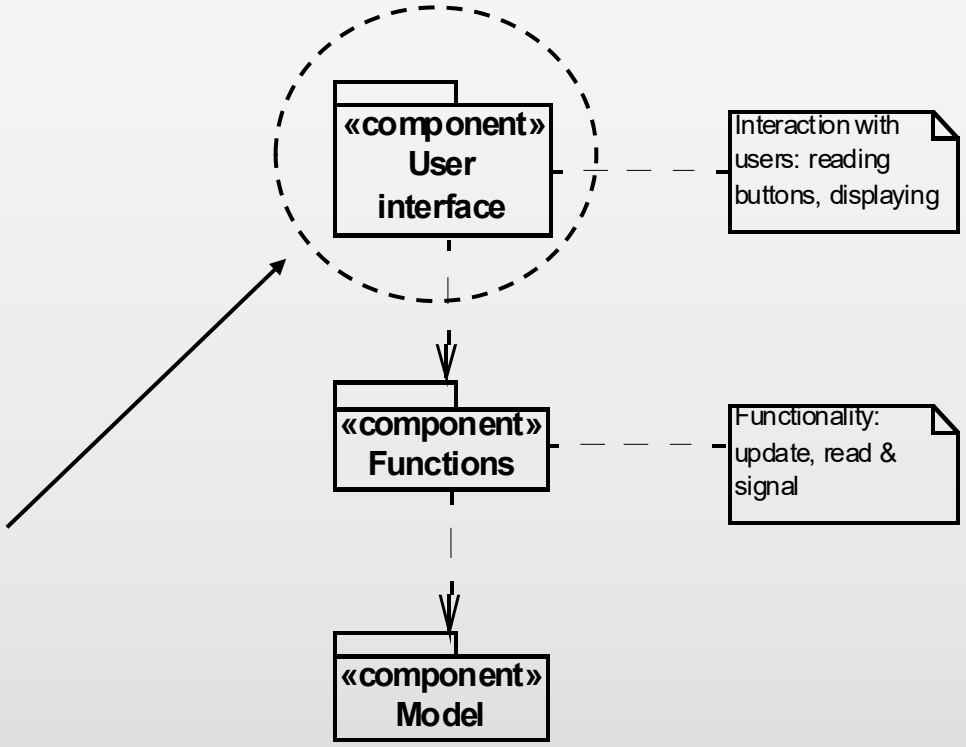
Components: Result



- A structural perspective
- Separates concerns in a system
- Emphasizes comprehensibility and flexibility

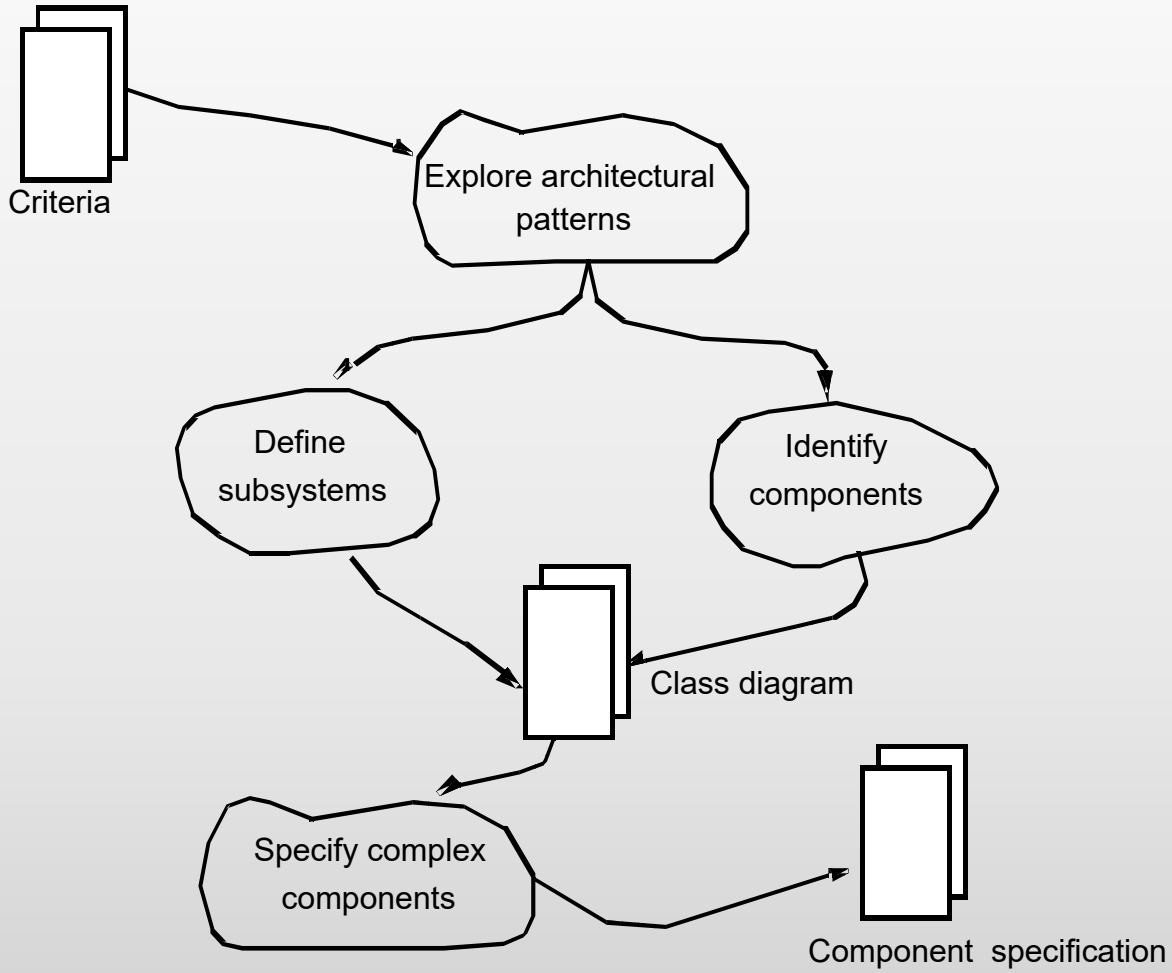
Key Concept: Component

- A collection of program parts
- Constitutes a totality
- Has a well-defined responsibility
- Smallest: a class
- Largest: a system

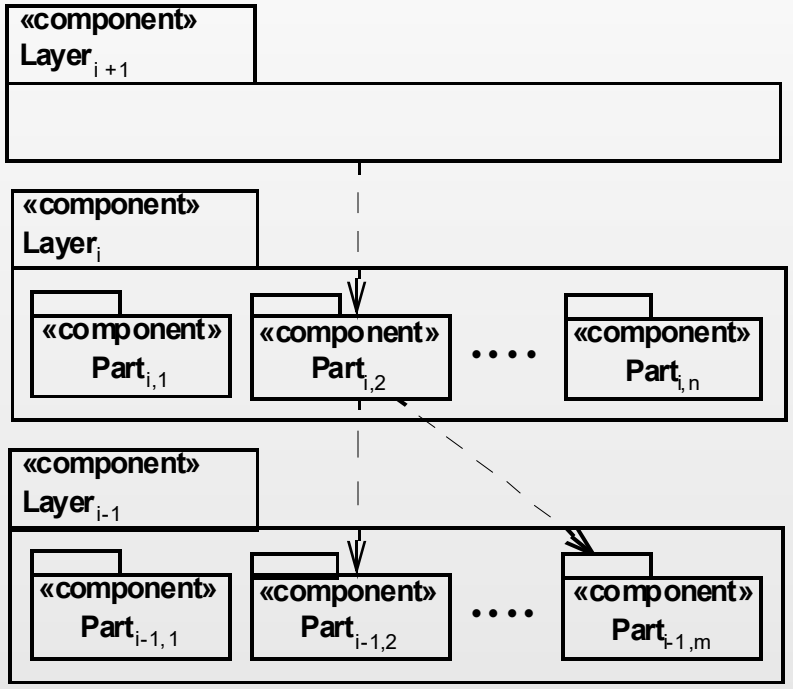


- Example:
  This component has the responsibility for reading the buttons and updating the display

Components: Activities

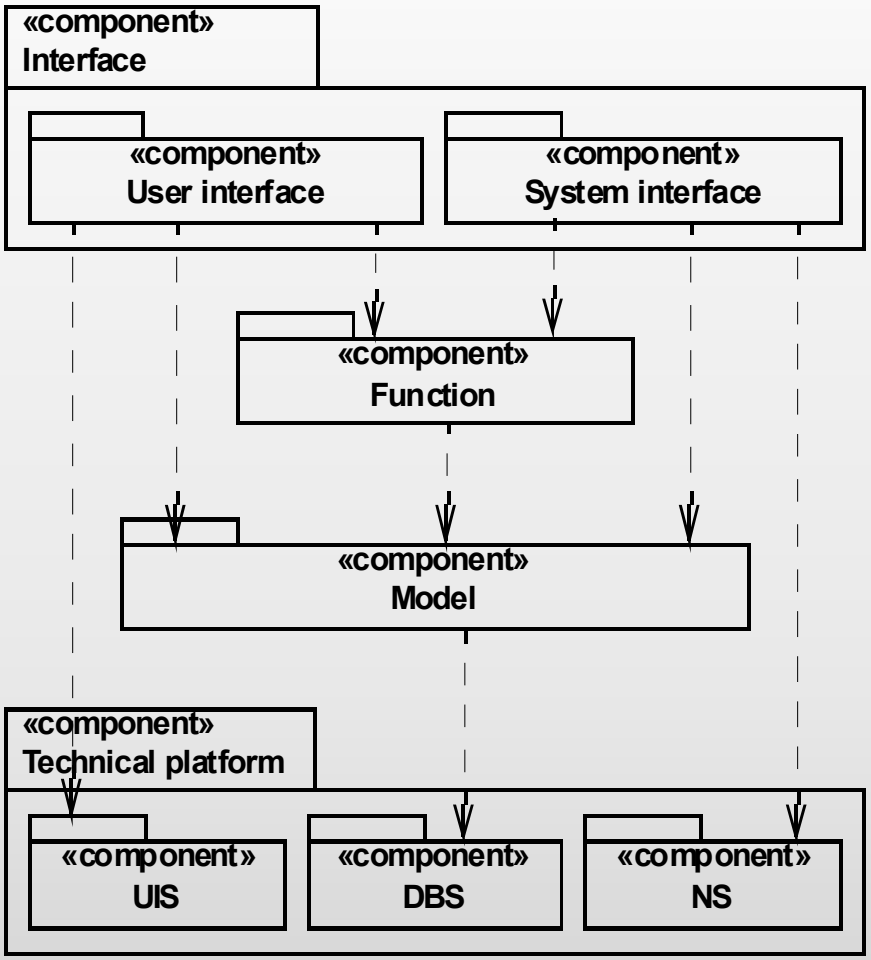


Pattern: The Layered Architecture



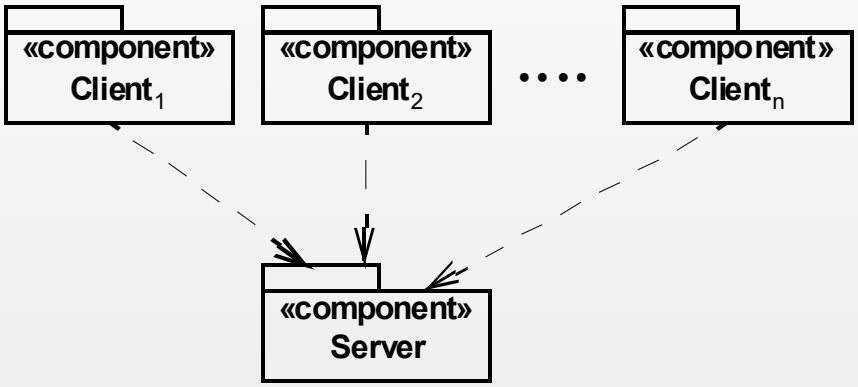
- Layer: describes a component's responsibility by the operations it provides to a layer above and those that are applied from the layer below
- Part: no substantial interaction with other parts in the same layer
- Closed architecture: only apply operations from an adjacent layer
  Open arcitecture: apply operations from any other layer
- Strict architecture: only apply operations from a layer below
  Relaxed architecture: apply operation from layer both above and below

Pattern: The Generic Architecure



- The generic architecture reflects the division of the context into problem domain and application domain
- "Technical platform" is an extension and encapsulation of the underlying technical platform
- Example: a single Dankort terminal

Pattern: Client-Server Architecture



Client	Server	Architecture
U	U + F + M	Distributed presentation
U	F + M	Local presentation
U + F	F + M	Distributed functionality
U + F	M	Centralised data
U + F + M	M	Distributed data

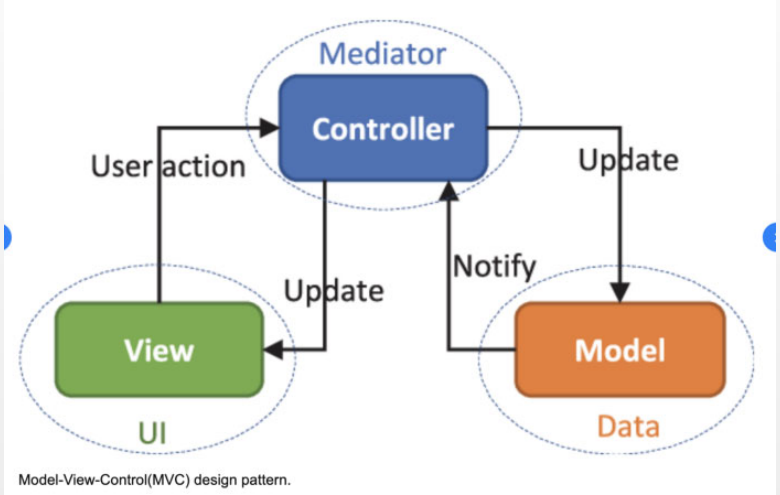
- Originally for distribution of physically (geographically) dispersed processors
- Can also be used logically, independently of processors
- One server and a number of clients
- Clients are assigned to the server dynamically
- The distribution can be based on various divisions between server and clients
- Example: the Dankort operator (Nets) and the shops

Pattern: Model-View-Controller (MVC)

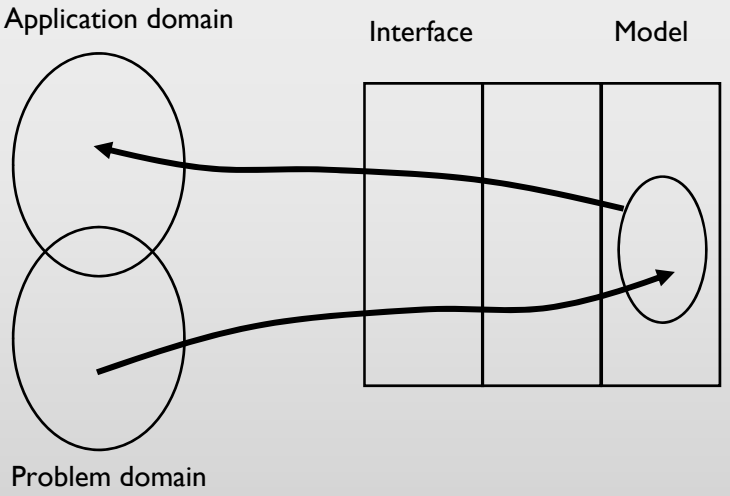
- Example: Excel with table and graphic

- https://medium.com/@rhodunda/mvc-design-pattern-fe76175a01de
- **Model:** contains all the logic for the application, including all the data for your program like databases, files etc.
- **View:** contains everything the user can see. This includes text, buttons, and anything else in the window.
- **Controller:** updates both the model and view. It accepts inputs and performs the corresponding update.

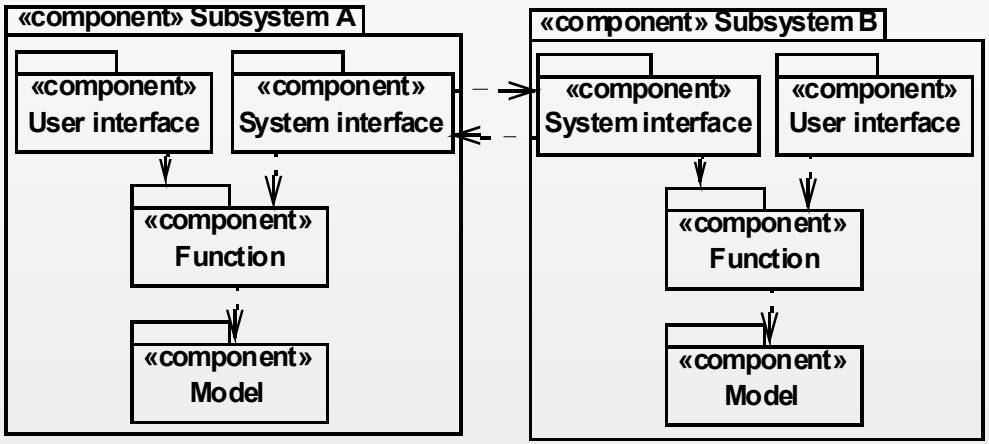
- Model = Model
- Views = objects in the UI (e.g. windows)
- What about System Interface objects that handle sensors? (not covered)
- Controller = object in the Interface that controls views and activates functions (that access the model)

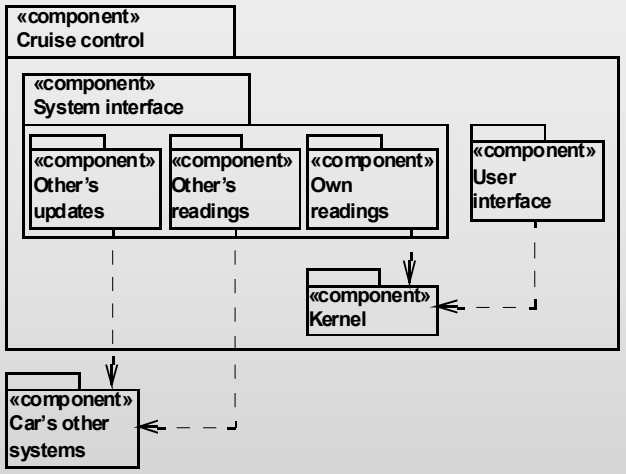


Model-View-Control(MVC) design pattern.



Define Sub-systems



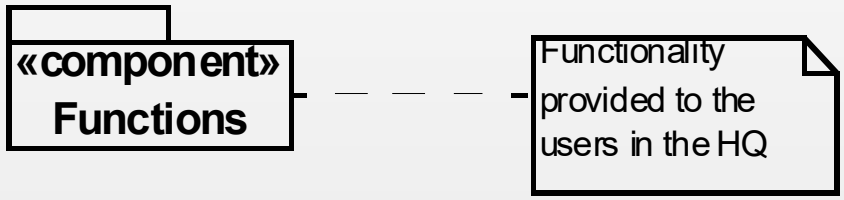


- Larger systems can be decomposed into several independent sub-systems
- Each sub-systems has its own architecture based on the generic architecture
- Example 1:
  - Cruise control
  - Car's other systems

  are two related sub-systems
- Example 2:
  - System and log-in system

Specify Complex Components

- Specify the component in detail by its
  - responsibility
  - dependency of other components
  - relation to the context
- In a schema or diagram



Components: Summary

Purpose	• To create a comprehensible and flexible system structure
Concepts	• Component architecture: a system structure of interconnected components. • Component: a collection of program parts that constitutes a whole and has well-defined responsibilities.
Principles	• Reduce complexity by separating concerns. • Reflect stable context structures. • Reuse existing components.
Result	• A class diagram with specifications of the complex components.