SCC.211 Software Design: Software Architecture

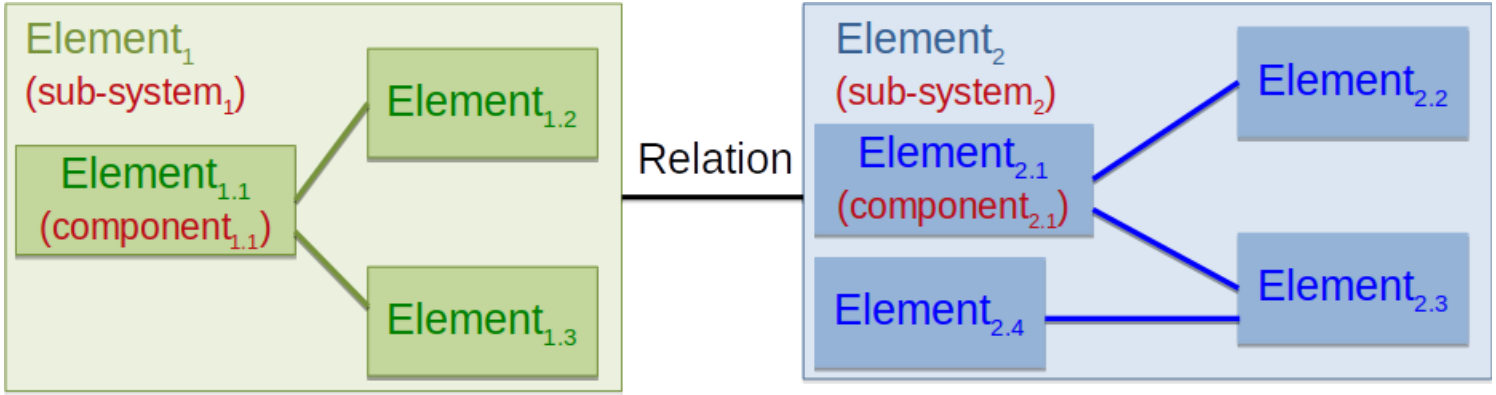
Professor Tracy Hall
Adapted from Jean Petric
Room C19, InfoLab21
School of Computing and Communications

Learning Outcomes

- Get introduced to the architectural level of a software product's abstraction
  - Understand how to partition a system into sub-systems and components
  - Understand "box and line" and "ball and socket" representations of a system
  - Be able to fully describe interfaces
  - Understand basic modelling of control between sub-systems

Software architecture

- The software architecture of a system is a set of structures needed to reason about the system, which comprise software elements, relations amongst and properties of both
- Software elements may be software sub-systems or components



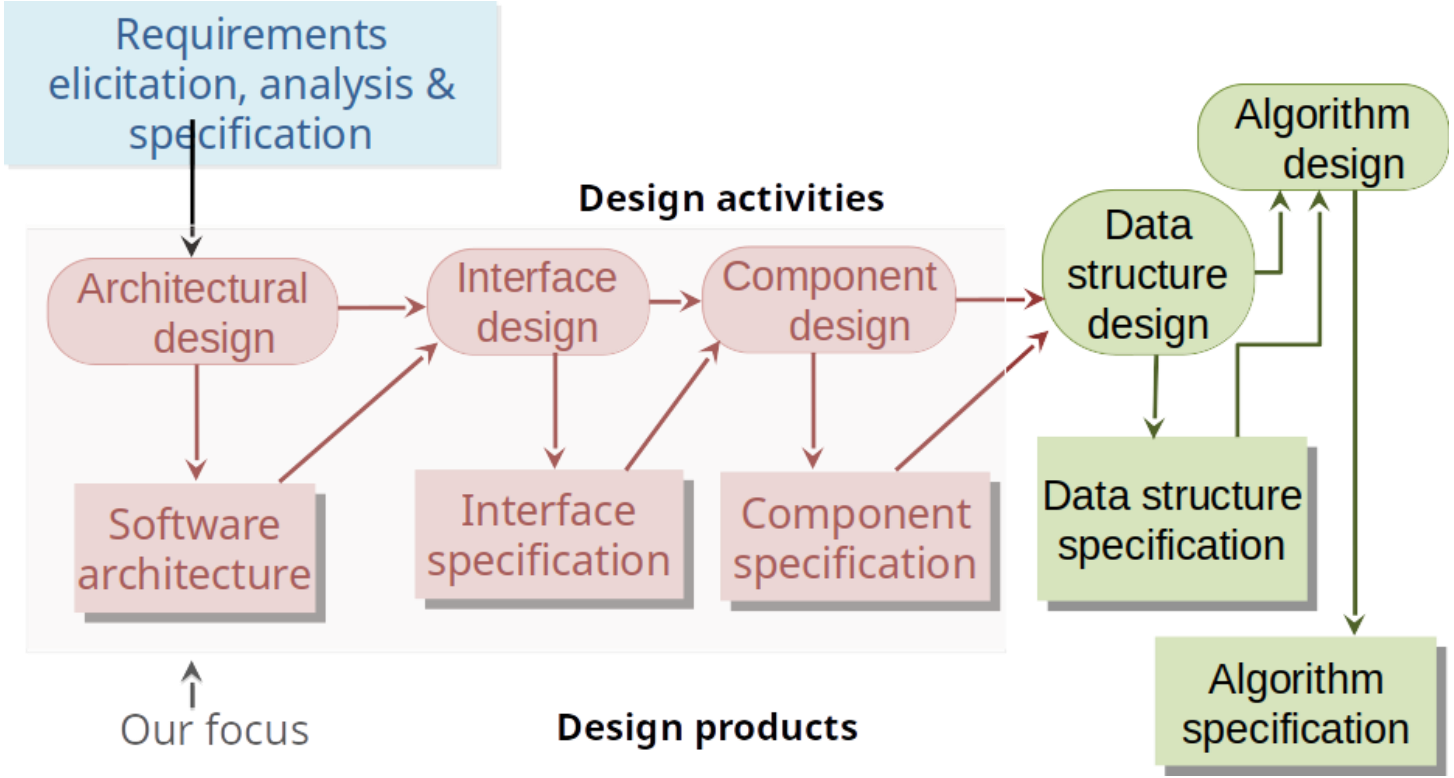
Relations indicated by links between elements

Implications of “architecture is a set of structures”

Lancaster University

- A structure is a simple set of elements held together by relations
- Architecture consists of structures and structures consist of elements and relations
- Architecture purposely omits certain information that is not useful for reasoning about the system

General software design process

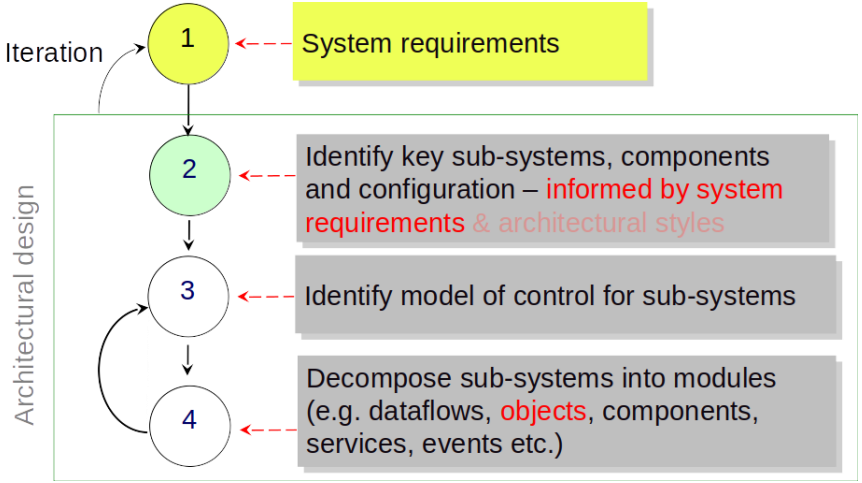


The software design process

- The design process may involve developing several models of the system at different levels of abstractions, i.e.
  - Architectural design - the sub-systems making up the system and their relationships are identified and documented. Model of control is identified.
  - Interface design - for each sub-system, it's interface with other sub-systems is designed and documented.
  - Component design - services (i.e. functional requirements) are allocated to different components and the interfaces of these components are designed
  - Data structure design - the data structures used in the system implementation are designed in detail.
  - Algorithm design - the algorithms used to provide services are designed in detail and specified.

Steps in architectural design

- The following activities are common to all architectural design processes:
  - System structuring - the system is structured into a number of principal sub-systems or components and the communication between sub-systems is identified.
  - Control modelling - a general model of control relationships between the parts of the system is established
  - Modular decomposition - each sub-system is decomposed into modules



Describing architecture

Formal notations

ComponentToConnectorLinks

$C2Link$

$components : \mathbb{P}\ C2Component$

$connectors : \mathbb{P}\ C2Connector$

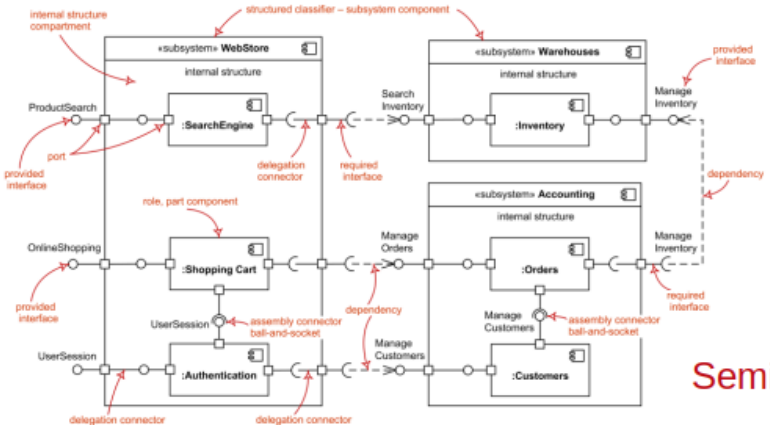
$\forall comp : components$

- $\exists_1\ conn1,\ conn2 : connectors;\ tport,\ bport : PORT \mid$
  $tport \in conn2.top\_ports$
  $\wedge\ bport \in conn1.bot\_ports$
  $\wedge\ conn1 \neq conn2$
  - $(comp.top\_port,\ bport) \in Link \wedge$
    $(comp.bot\_port,\ tport) \in Link$

Architecture Description Languages

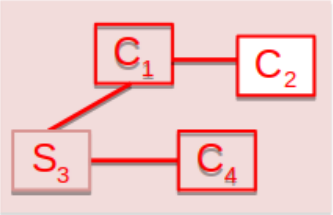
architecture ::= [comment] {element | component}

element ::= defined_type | port | connector | role
component ::= subtype_component | component_instance
| concrete_component



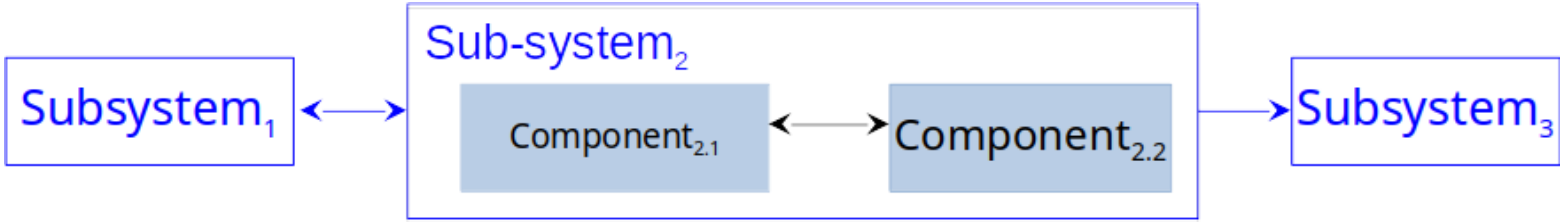
Semi formal

Box and Line notation



Architectural representation: box and line diagram

- At its most abstract level, an architectural design may be depicted as a block diagram (using box and line diagram):
  - The block diagram comprises the system's major sub-systems, their components and relations
  - Sub-systems are often identified by clustering logical functionality (e.g. UI, computation, management, storage, etc.)
  - Boxes within boxes indicate that the sub-system itself has been decomposed to components (we often keep decomposition to two levels)
  - Links represent function calls, data or control signals that are passed from component to component in the direction of the arrow



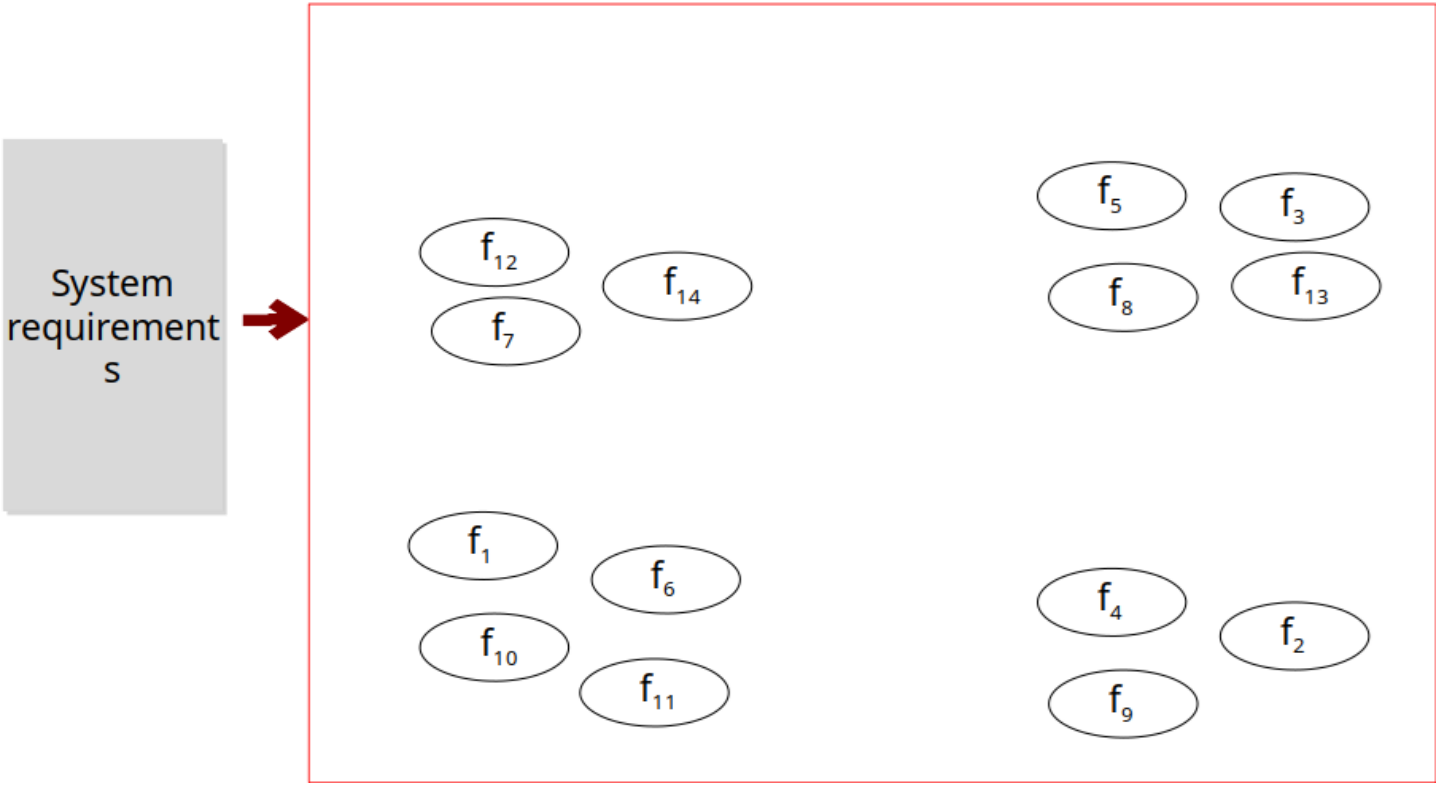
Identifying sub-systems & their interfaces

1. Identify major functions the system is required to provide
   a. These can be distilled from textual requirements and use cases
2. Associate the major system functions with sub-systems by grouping together logically similar functions
   a. E.g. functions concerned with specific system tasks, performance-critical functions or security-critical functions etc.
3. Sub-systems may be decomposed further using the same principle
   a. Limit the system decomposition to two levels
4. Specify sub-system interfaces

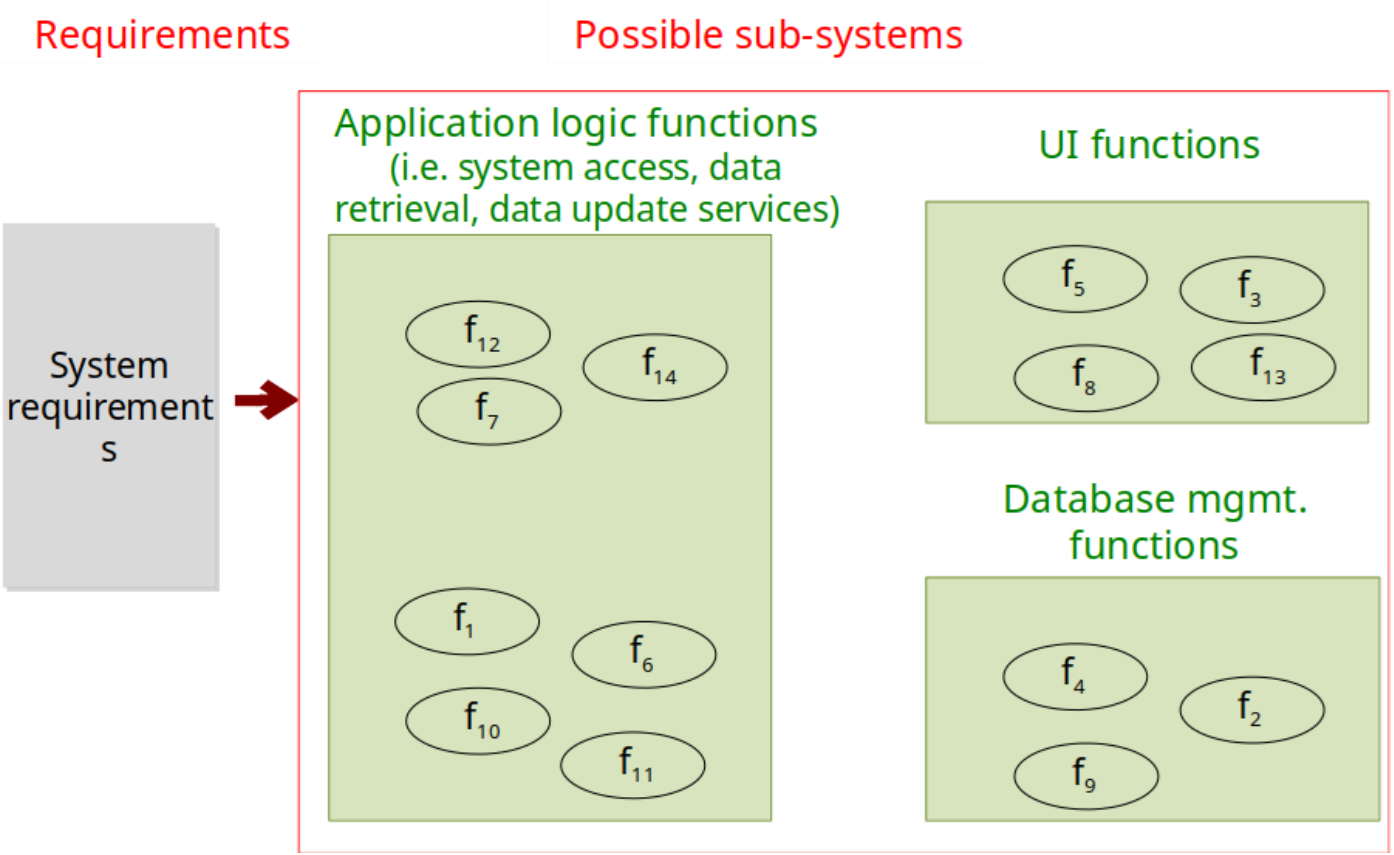
Identifying sub-systems: generic example
Partitioning requirements into sub-systems

Requirements

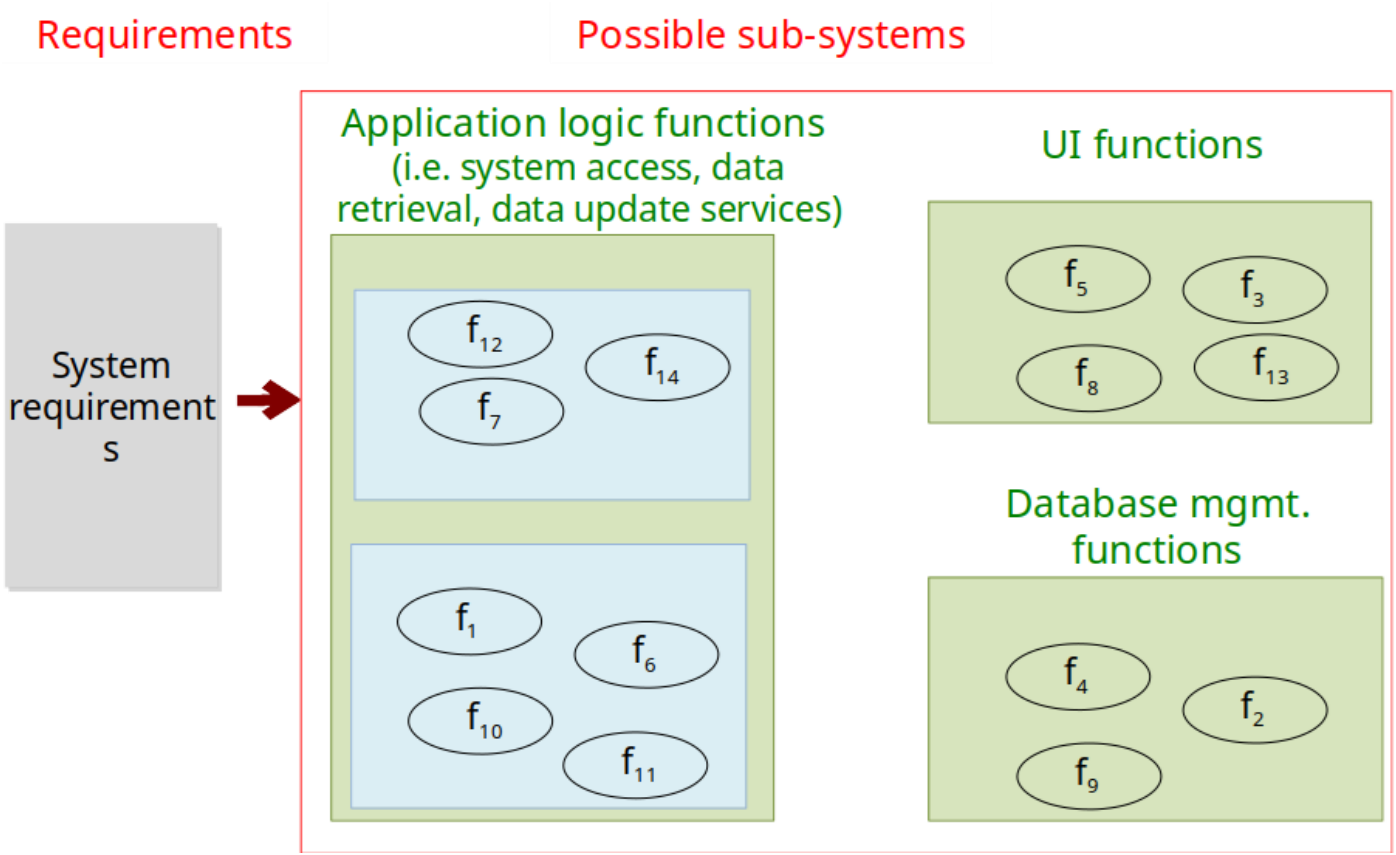
System functions



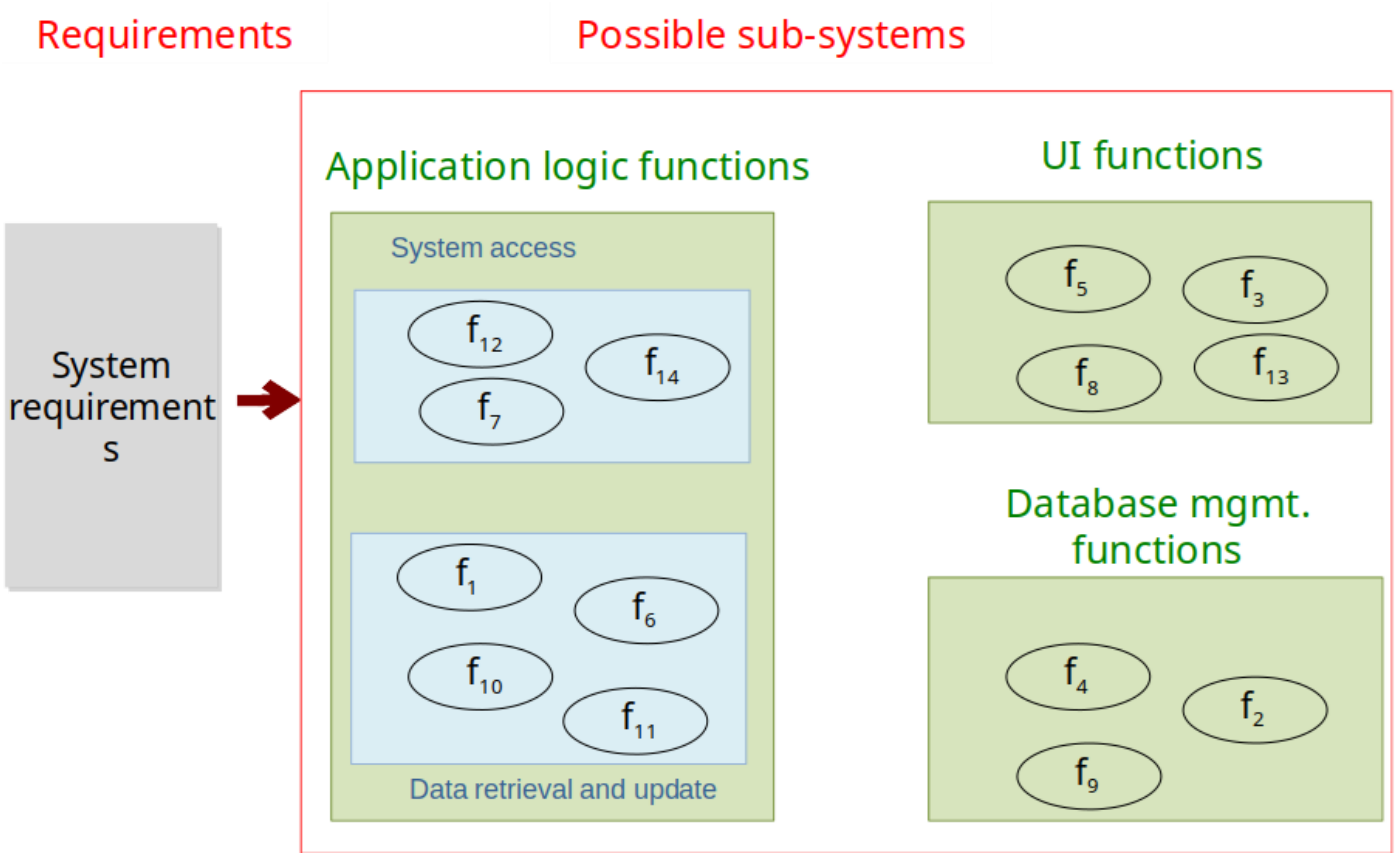
Identifying sub-systems: generic example
Partitioning requirements into sub-systems



Identifying sub-systems: generic example
Partitioning requirements into sub-systems



Identifying sub-systems: generic example
Partitioning requirements into sub-systems



Alarm system example

- Consider an intruder alarm system that activates a siren and notifies the local control centre when its sensors are tripped. The alarm system has:
  - Movement (motion), door and window sensors
  - A video system that is activated when potential intruders are detected. The date, time, property address and video of potential intruders is relayed to an external control centre
  - A messaging system that sends a text message together with a preconfigured number of still images of potential intruders to the home owner
- Task
  - Identify possible sub-systems for the alarm system and their interaction. **Hint**: Group together related functionality to form sub-systems

Alarm system: identifying sub-systems

I. Read through the system description to establish what the system is required to do OR what the system does
II. List the system functions
III. Group related functionality
IV. Associate functionality of sub-systems
V. For each sub-system establish if there is need to partition further

_____ system functions _____

- Consider an intruder alarm system that **activates a siren** and **notifies** the local control centre when its **sensors are tripped**. The alarm system has:
  - Movement (motion), door and window sensors
  - A video system that is **activated** when potential intruders are **detected**. The date, time, property address and video of potential intruders is **relayed** to an external control centre
  - A messaging system that **sends** a text message together with a preconfigured number of still images of potential intruders to the home owner

Alarm system: identifying sub-systems

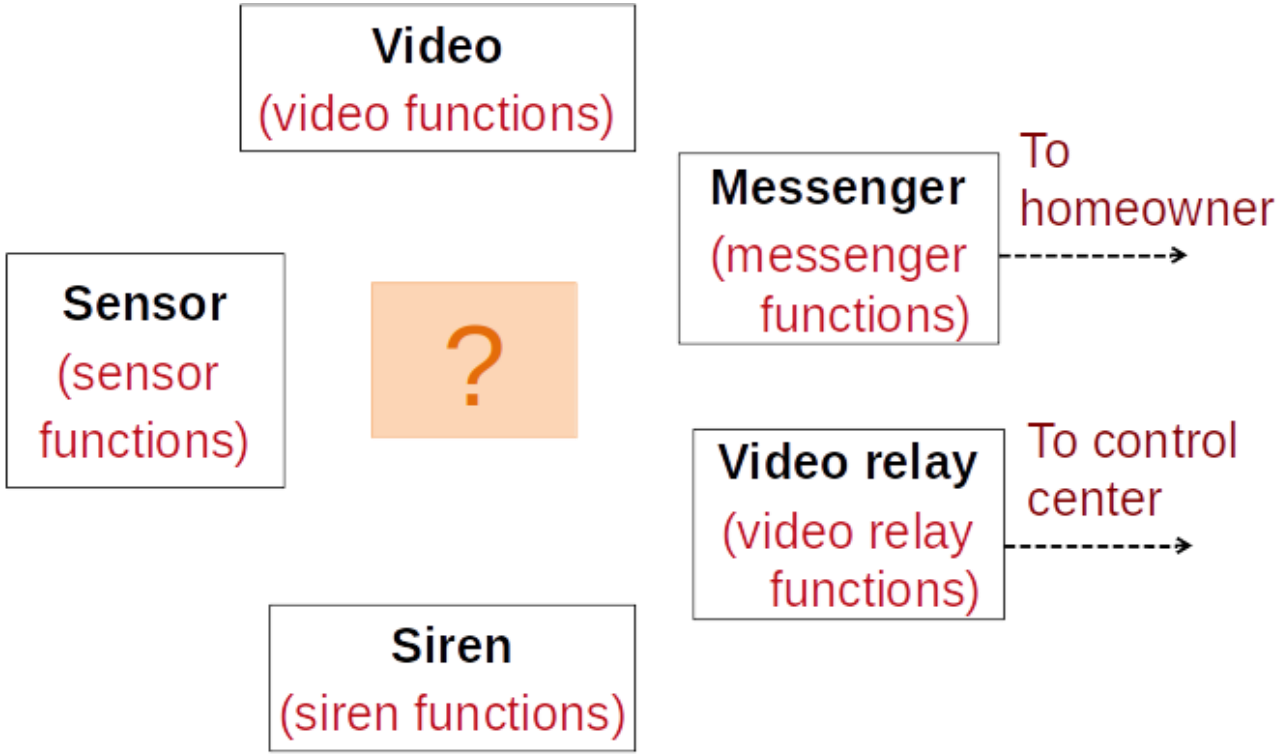
I. Read through the system description to establish what the system is required to do OR what the system does
II. List the system functions (**sensing intruders, activating siren, notifying local control, activating video, messaging home owner**)
III. Group related functionality (**sensor, siren, video relay, video, message**)
IV. Associate functionality of sub-systems
V. For each sub-system establish if there is need to partition further

_____ system functions _____

- Consider an intruder alarm system that **activates a siren and notifies** the local control centre when its **sensors are tripped**. The alarm system has:
  - Movement (motion), door and window sensors
  - A video system that is **activated** when potential intruders are **detected**. The date, time, property address and video of potential intruders is **relayed** to an external control centre
  - A messaging system that **sends** a text message together with a preconfigured number of still images of potential intruders to the home owner

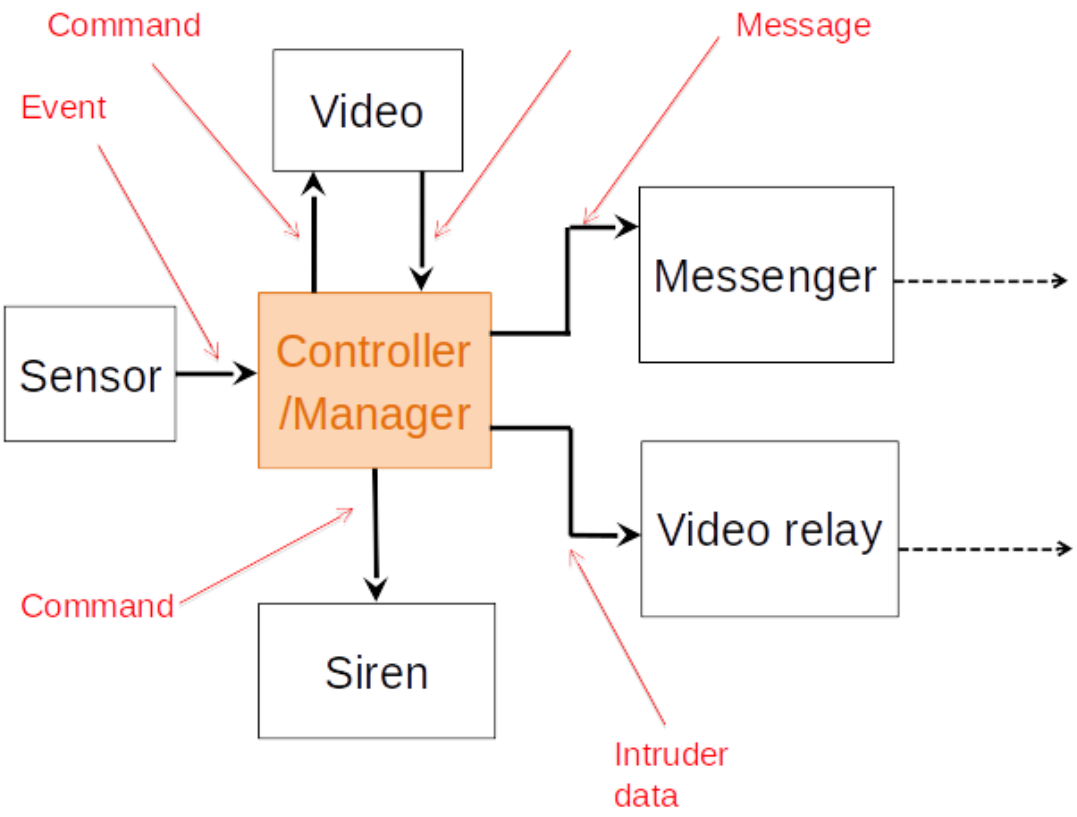
Alarm system: identifying sub-systems

- Link sub-systems that interact with each other (i.e. those that exchange data or control information)
- Point your arrow in the direction of the data or control flow.

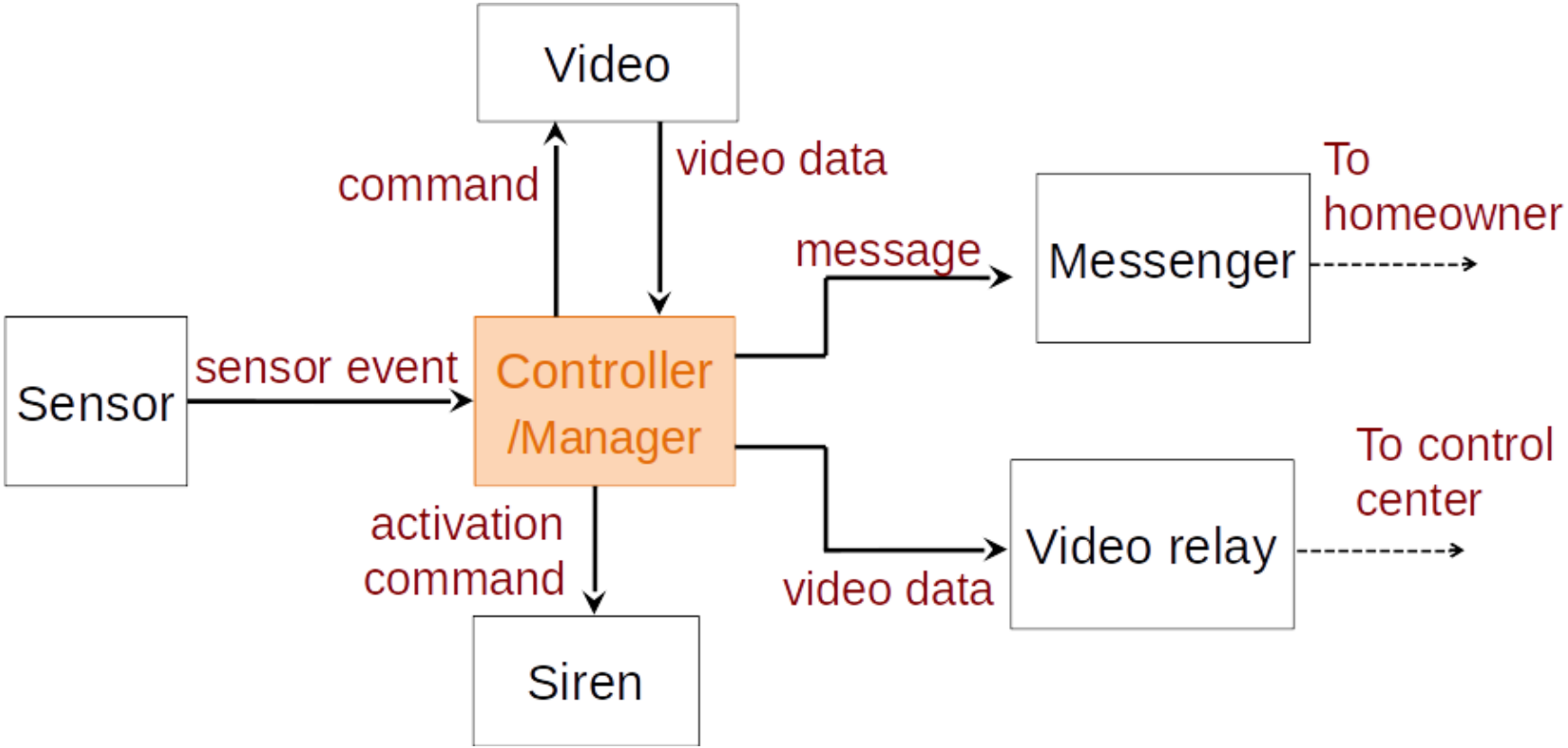


Alarm system: identifying sub-system interaction (adding “glue” component, i.e. “Controller”)

- Link sub-systems that interact with each other (i.e. those that exchange data or control information)
- Point your arrow in the direction of the data or control relation.



Alarm system: label calls & control flows



Specifying sub-system interfaces

- The description of a sub-system or component interface should include the following elements:
  - Interface name: a unique identifier for the interface (e.g. ISensor)
  - Operations: the name of each operation, together with input parameters and output/return, e.g. motionDetected(:int)
    - Operation name: motionDetected
    - Input: None
    - Output/return: int (indicating location of sensor or sensor number)
  - Exceptions: the name and data context for operation exceptions (i.e. what causes the exception, e.g. invalid input parameter)
  - Service quality (optional): non-functional properties associated with the service provided at the interface

Specifying sub-system interfaces

- An interface may have several operations, each with its own input and output parameters
  - An output parameter represents what is returned by an operation
- For example, the **ISensor** interface may have the following operations:
  - motionDetected() :int
  - windowOpened(): bool
  - doorOpened(): bool
- **I** is convention used to denote “Interface”: used less and less
- Common alternatives to **I** are names that end in **er** or **able**, e.g. Reader, Writer, Executer, Moveable, Destroyable, Closeable etc.

Specifying sub-system interfaces: ball and socket

Lancaster University

provided

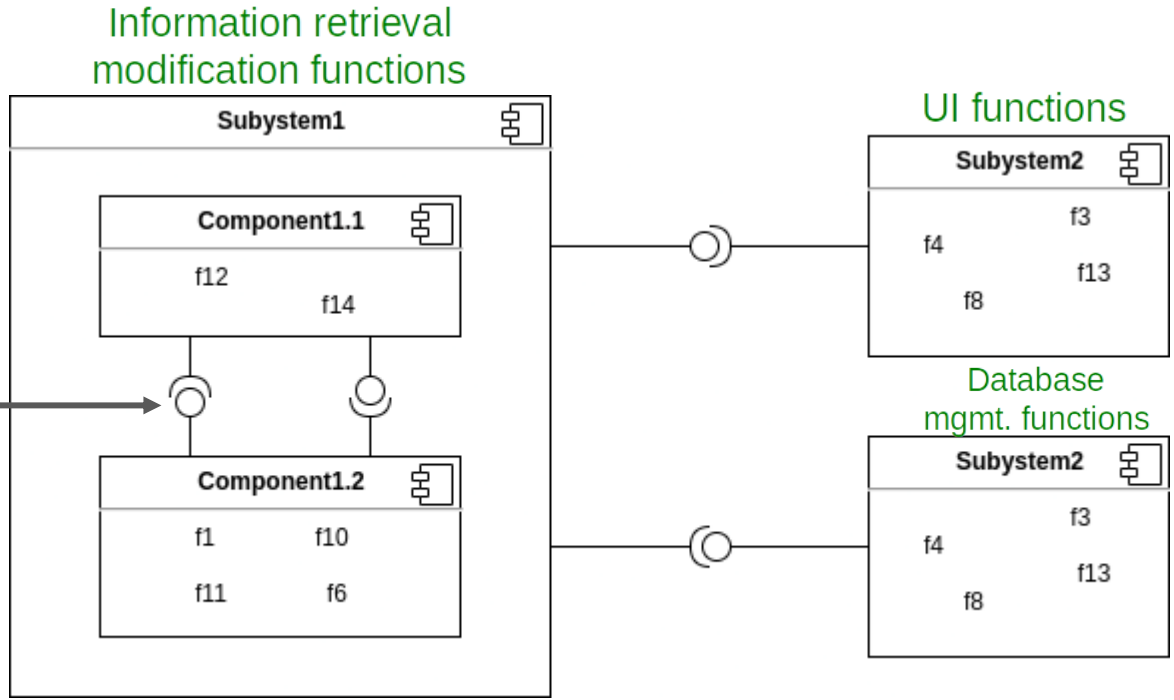
(a service provider)

required

(a service user or caller)

Description

- Interface name
- Operations
- Input parameters
- Output/Return parameters
- Exceptions



Guidelines for interfaces

- Service provider implements all the operations defined in its interface, e.g.

Service Provider

> Performs computation
> Manages data (DB)
> Verifies requests
> Processes a request

Event Subscriber

>Implements event processing routine/ event handler

- Service user calls operations provided by the interface, e.g.

Service Requester

> Compute this for me
> Store this data for me
> Verify this for me
> Process this for me

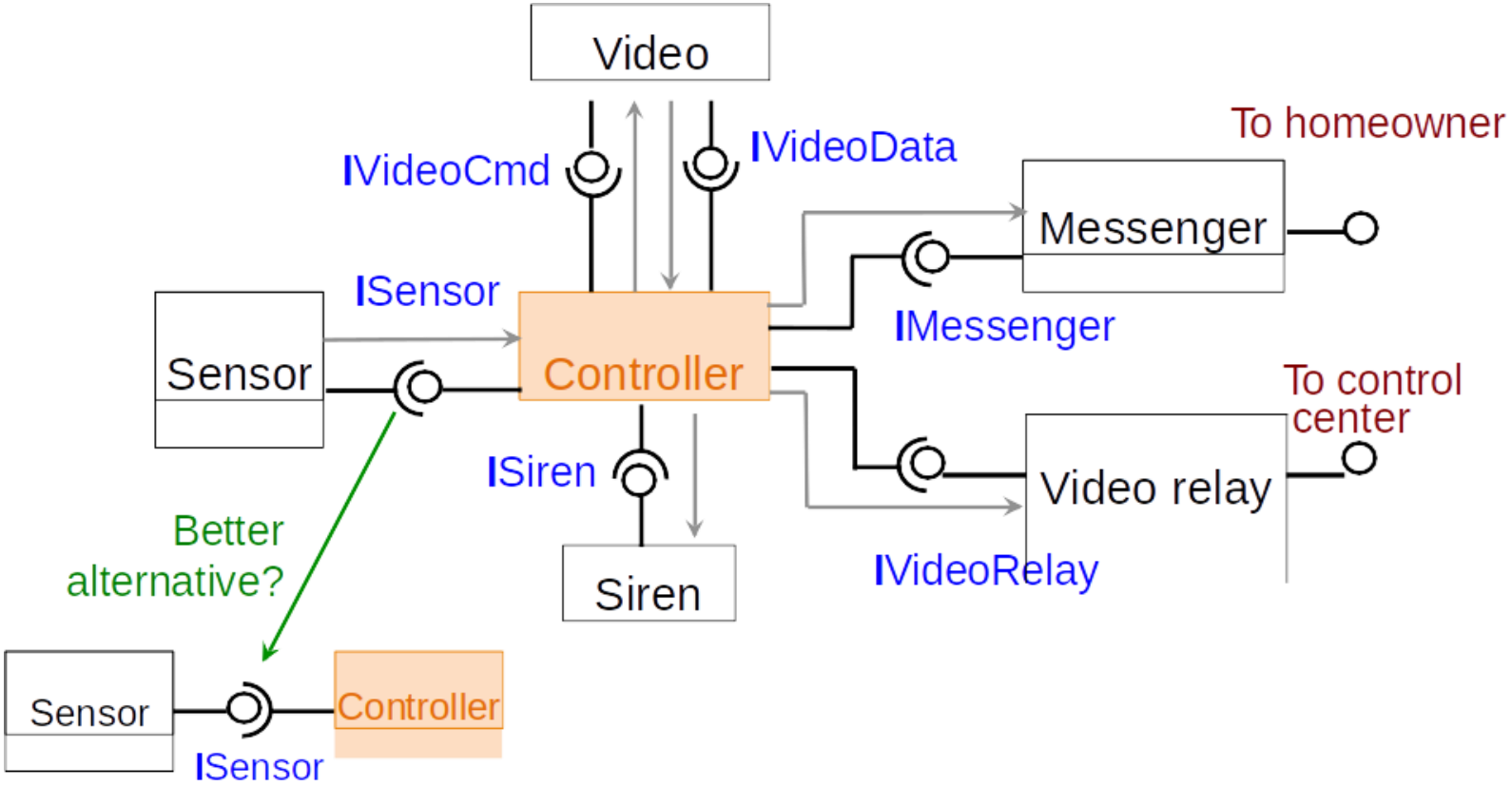
Event Publisher

> Process this event/ handle this event

provided
(a service provider)

required
(a service user or caller)

Alarm system with interfaces (ball and socket)



Interface operations and parameters

Interface	Operation	input	Output/Return	Exceptions
ISensor	motionDetected	none	:int	inputDataException //invalidInputData
	doorOpened	none	:boolean	inputDataException //invalidInputData
IVideoCmd	activate	none	:boolean //true or false	none
	deactivate	none	:boolean //true or false	none
IVideoData	getData	none	:time, :date, :address, :mp4	none
IMessenger	sendMessage()	:message	none	messageException //invalidMessage
ISiren	activate()	none	none	none

Wrap up

- Architectural level is an abstraction that is above implementation and below software requirements
- Software design process entails several stages at the architectural level:
  - Identifying key sub-systems, components and configuration
  - Identifying model of control for sub-systems
  - Decomposing sub-systems into modules
- "Box and line" and "Ball and socket" diagrams help us represent architecture in a standard way