Supervision Session 4
Example Solution for Assignment 1
Launch A1T2

Truong Ho-Quang
truongh@chalmers.se

Supervision Session 4 Topics

- Preliminary Feedback: Task 1, Assignment 1
- An example solution of Assignment 1
- Launch of Task 2 – Implementation

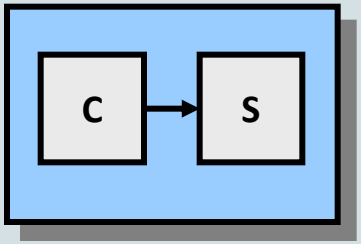
Preliminary feedback on Assignment 1

I have seen:

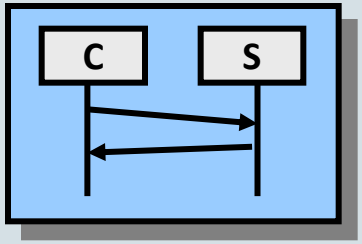
- Lack of consistency:
  - Between structural view and sequence diagram and deployment diagram.
- Too abstract/coarse grained components
  - If someone reads the architecture (documents), but not the requirements, they must be able to understand what the system does.

Regular Client-Server

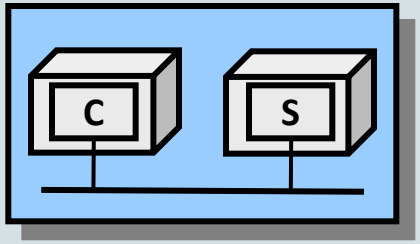
Structure Diagram
Component/Package/Class



Sequence Diagram



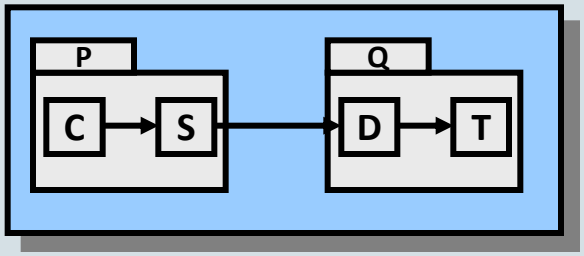
Deployment Diagram



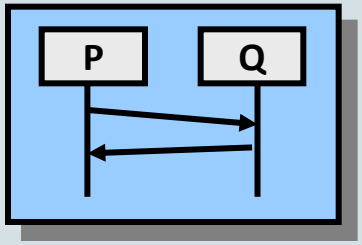
Regular Client-Server

You can use packages as a unit of communication and deployment

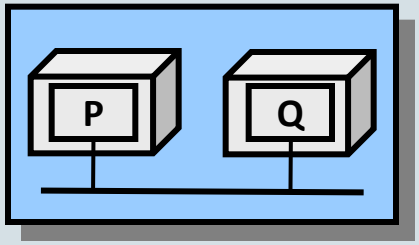
Structure Diagram



Sequence Diagram

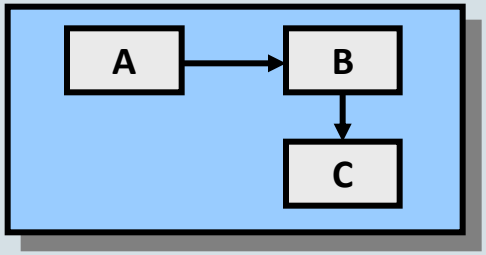


Deployment Diagram

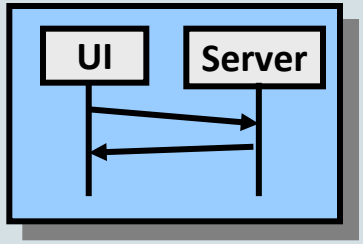


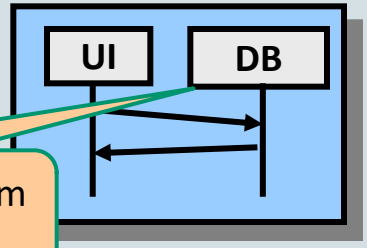
Not Consistent:

Structure Diagram



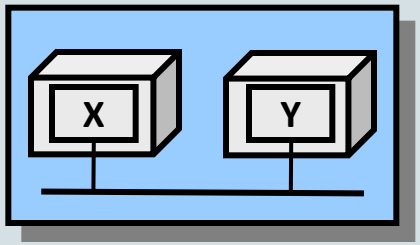
Sequence Diagram





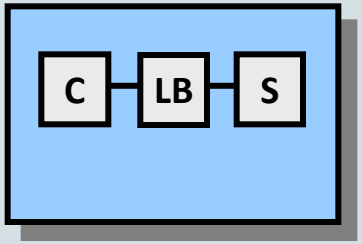
DB is not in the Structure diagram
Also, naming too general!

Deployment Diagram

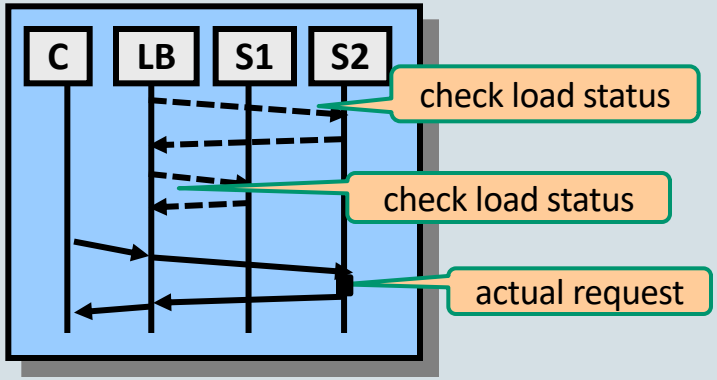


Client-Server with load balancer

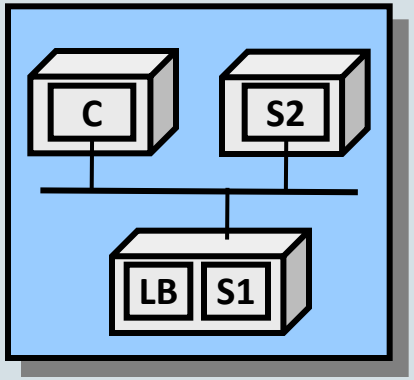
Structure Diagram



Sequence Diagram



Deployment Diagram



Many variations are possible

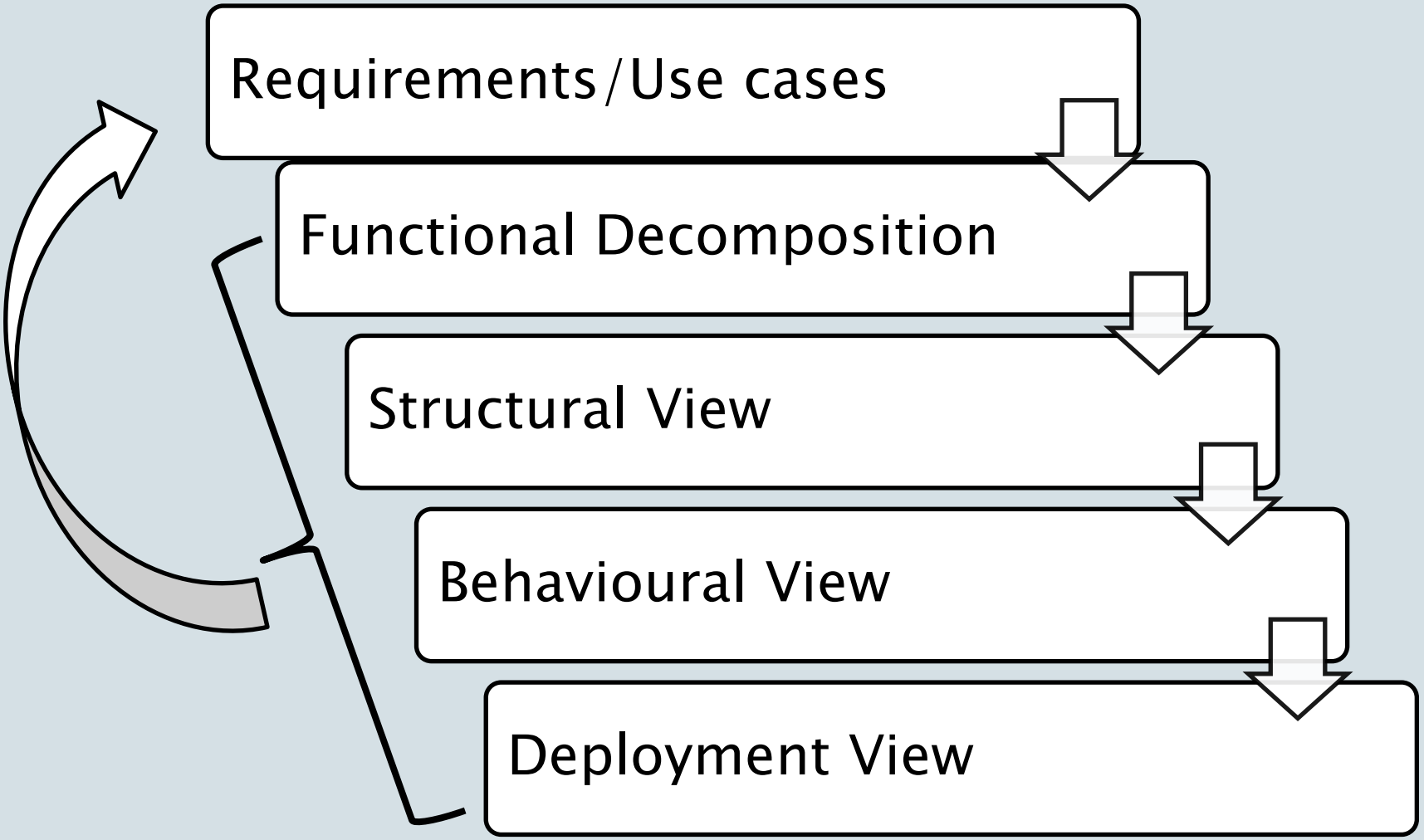
(Almost) the same tactic can also be used for improving availability

EXAMPLE SOLUTION FOR ASSIGNMENT 1

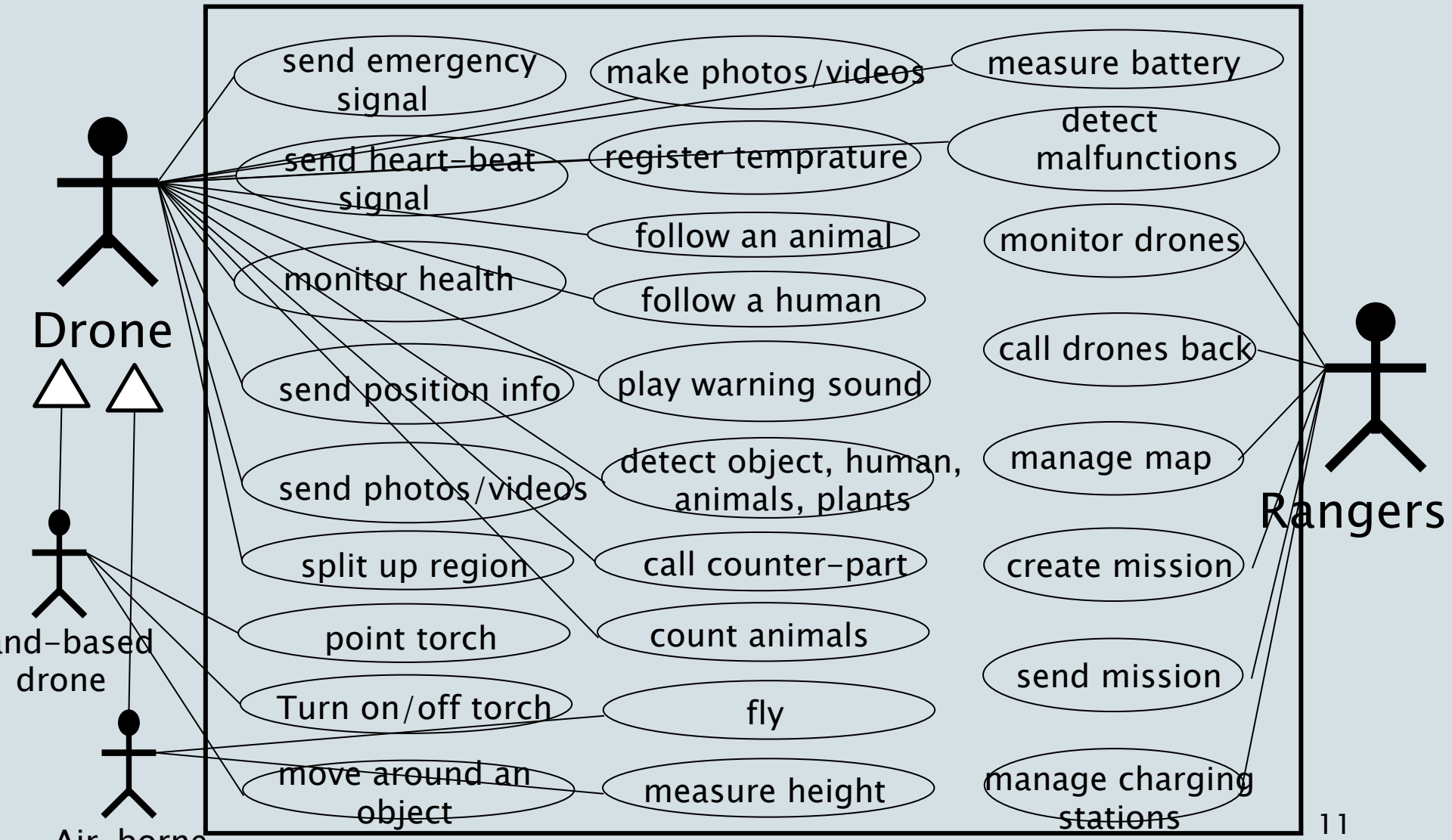
General Approach

- Physical units on which software runs
  - Unmanned Drones
  - System at the Ground Station
  - Servers, Network, AI, … } Depends on solutions
  - Charging Stations } Less important
  - Heart-beat Tags

Approach


Requirements/Use cases
Functional Decomposition
Structural View
Behavioural View
Deployment View

Usecases(*)



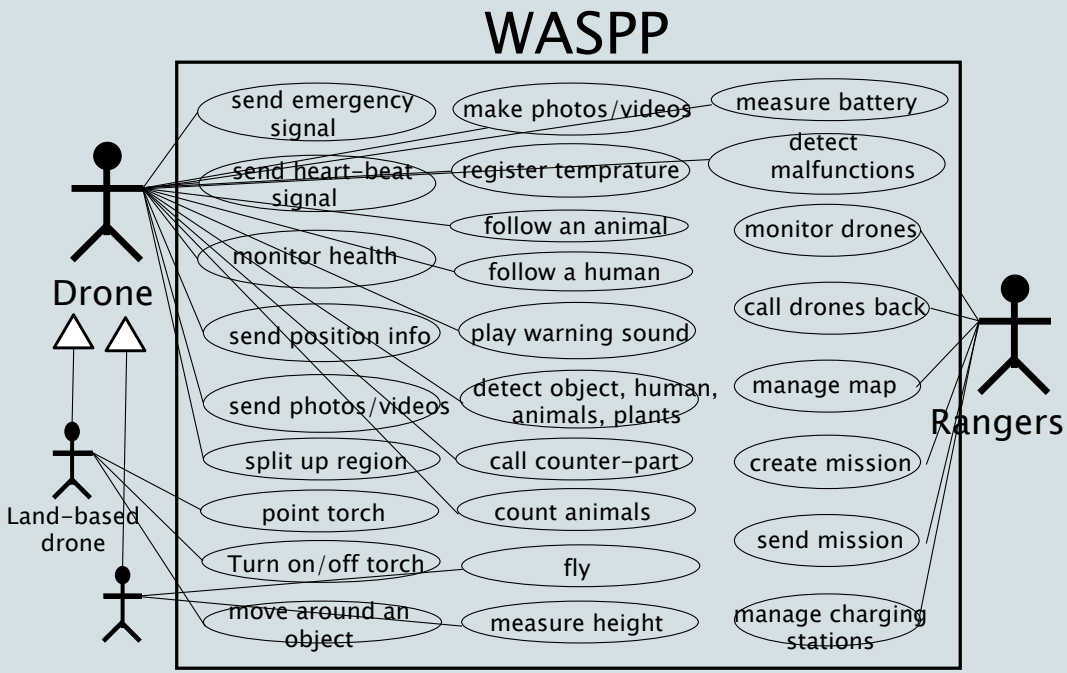
(*) This is not a complete usecase diagram (no include, exclude). The purpose is to show all functionalities of the system

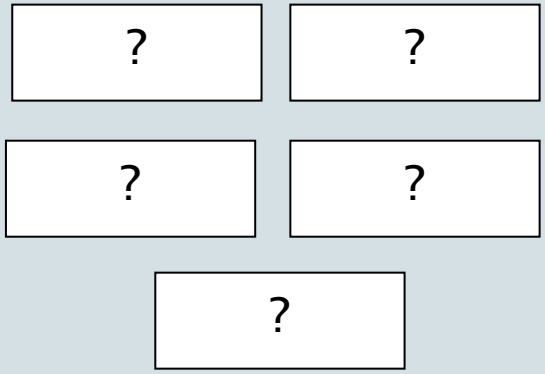
Functional Decomposition

- Input: Func. Requirments, Usecases
- Output: Functional Decomp. Diagram
- Approach
  - Think of system's functionality
  - Show 'what' system does

- Common mistakes
  - Think too much about solutions
  - Show 'how' system does that

Functional Decomposition Diagram – How to?





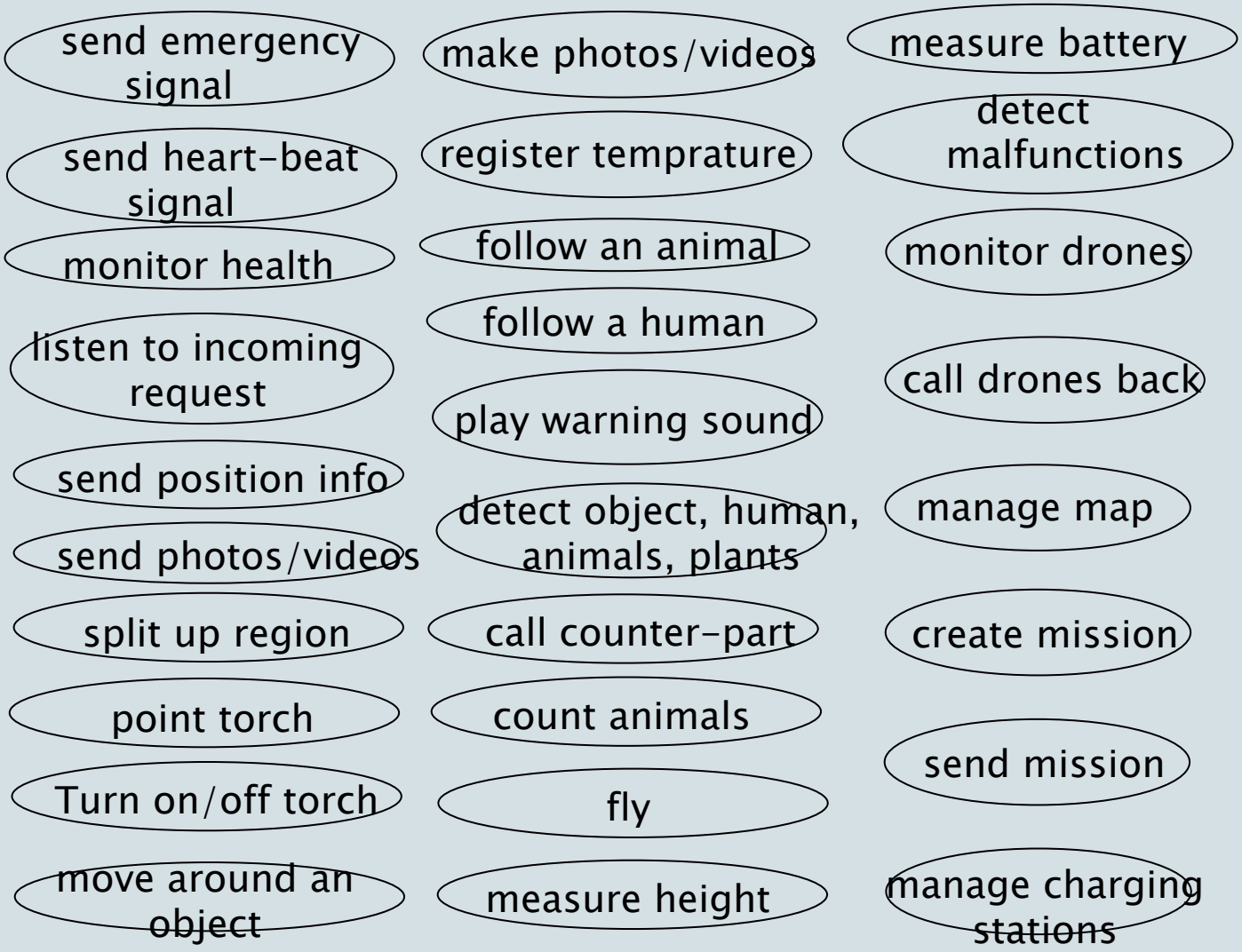
Functionalities

Functionality Areas

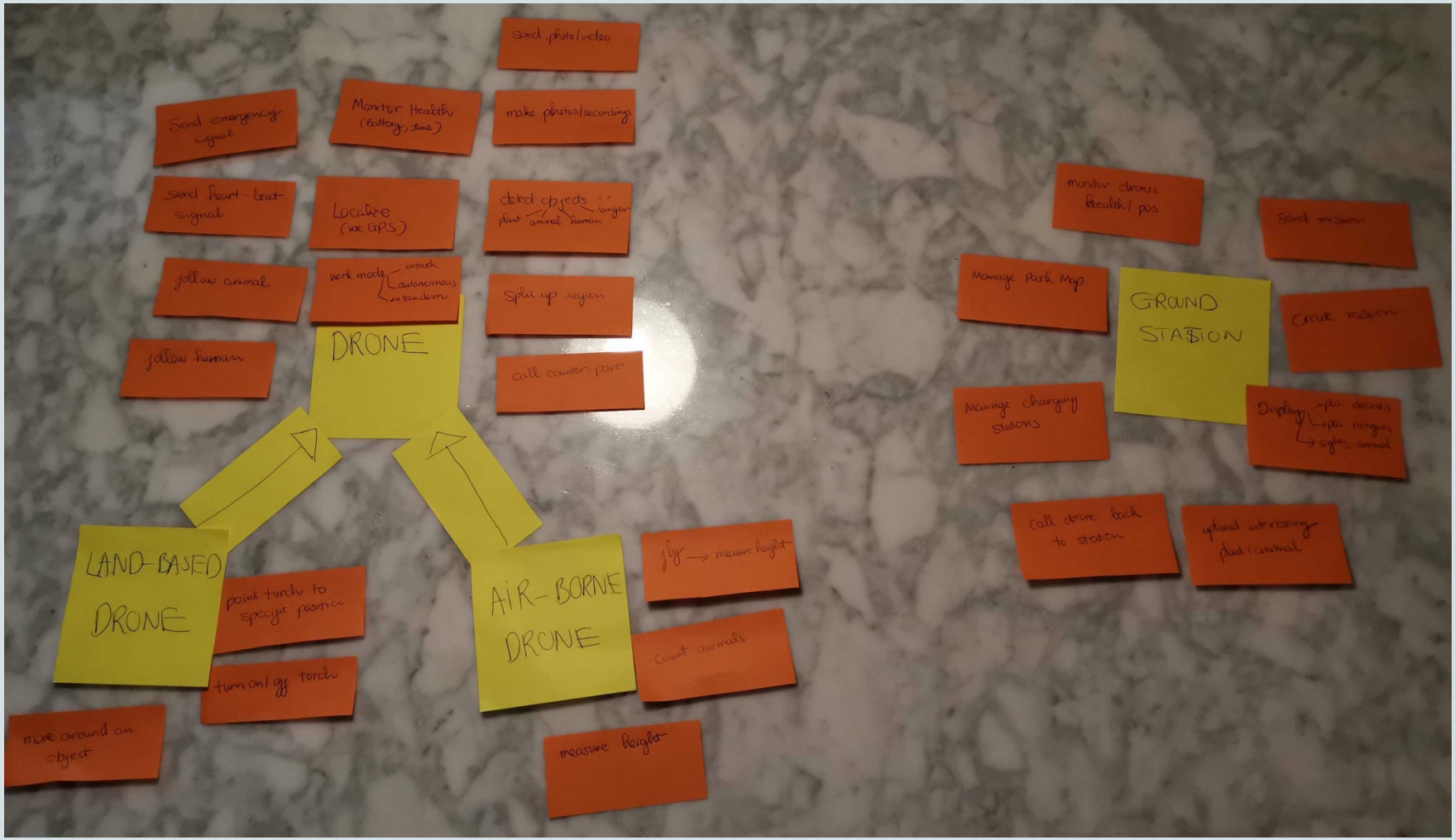
A Functional Decomposition Approach

- My approach
  - Consider Drone and Ground Station as two main **functional sub-systems**
  - Aims: To achieve better separation of functionalities between the two sub-systems
- Other approaches (also correct)
  - Do NOT view Drone and Ground Station as **functional** sub-systems
  - Pitfalls:
    - Be careful with “same” functions
    - E.g. (Ground Station) monitors drone’s health <> (drone) measures its health. Both can be categorized under “Drone Management”.

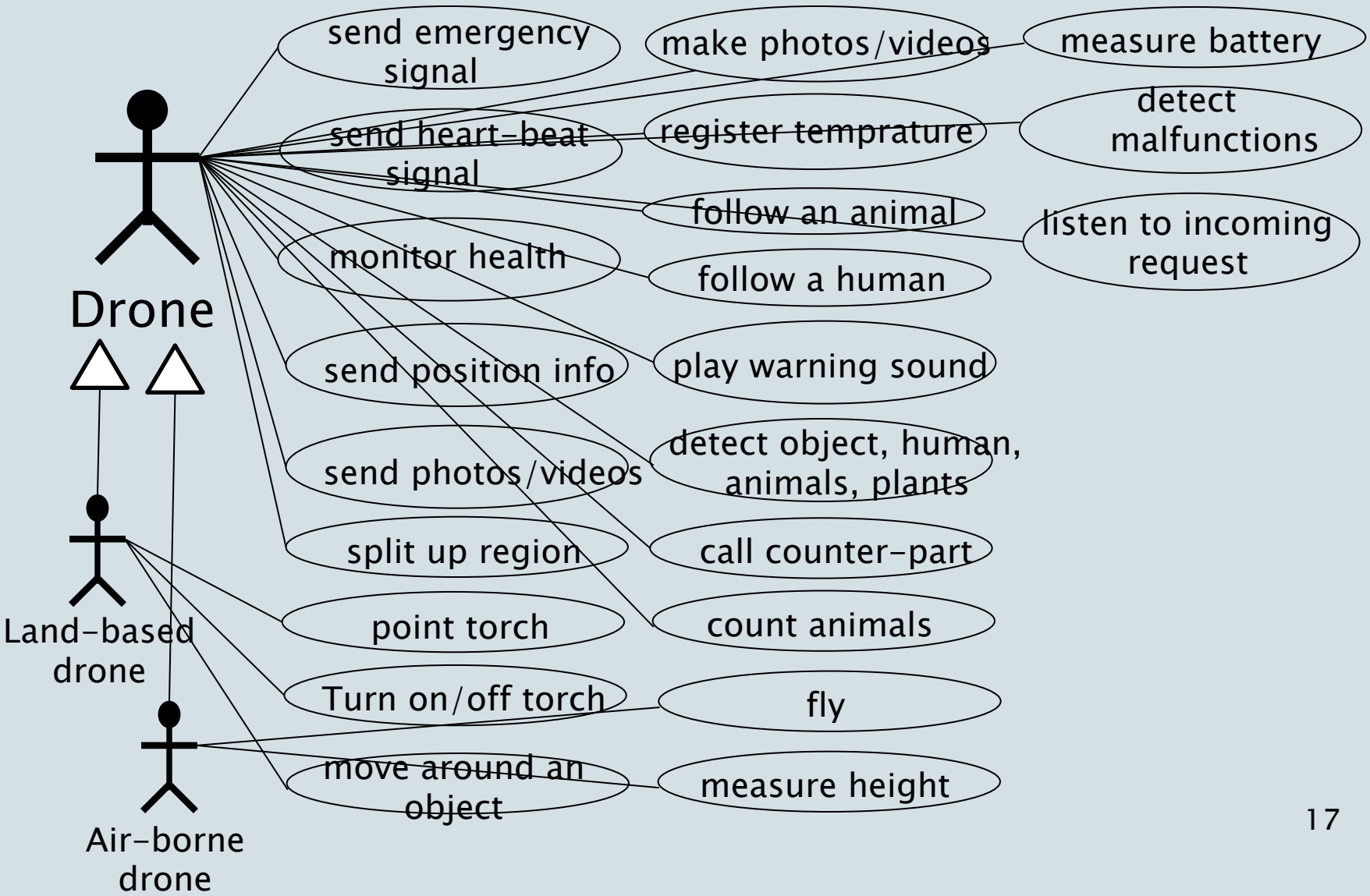
Functionality pool



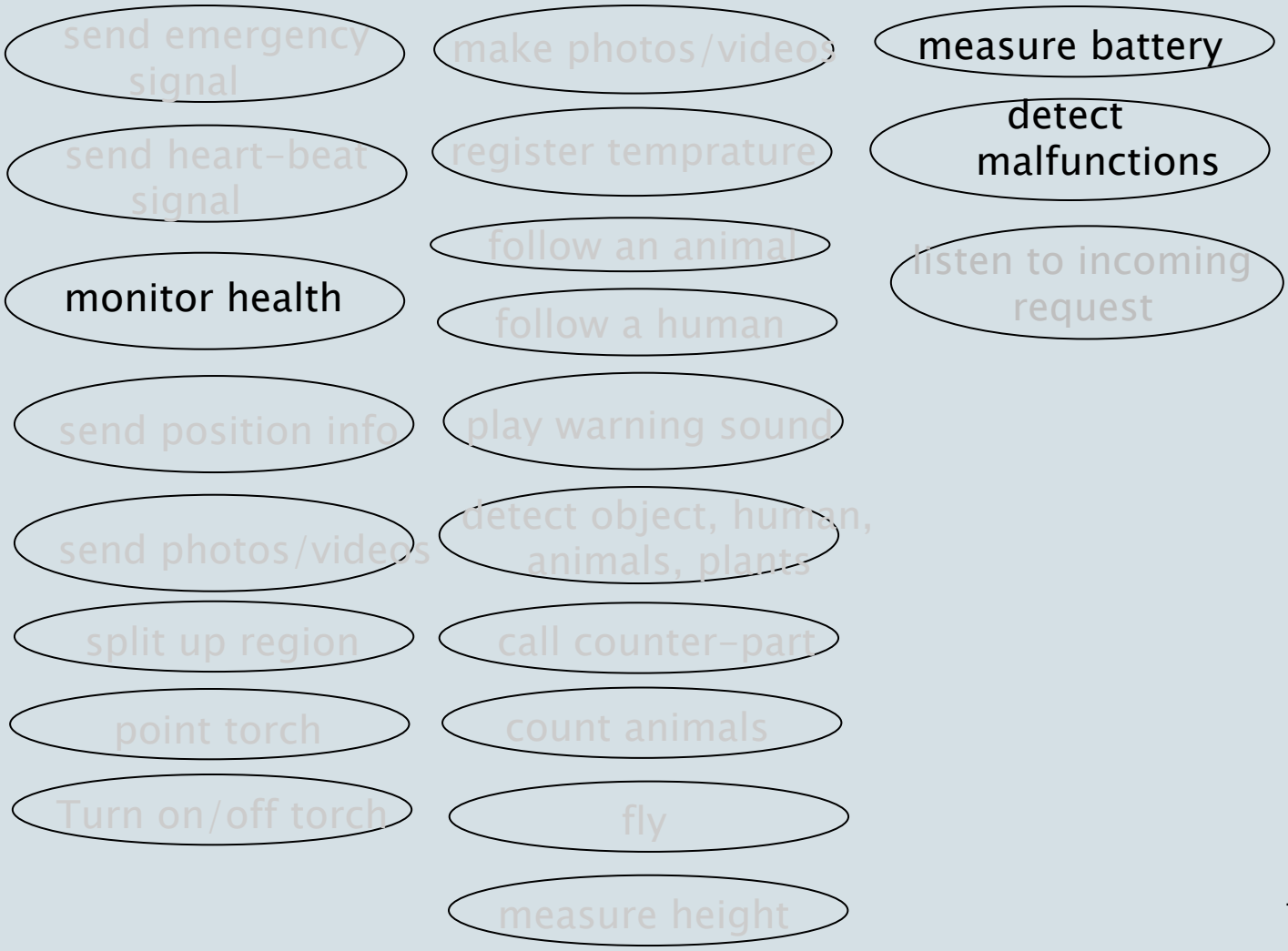
Group work: Use post-it notes



Drone Sub-system's Functionality

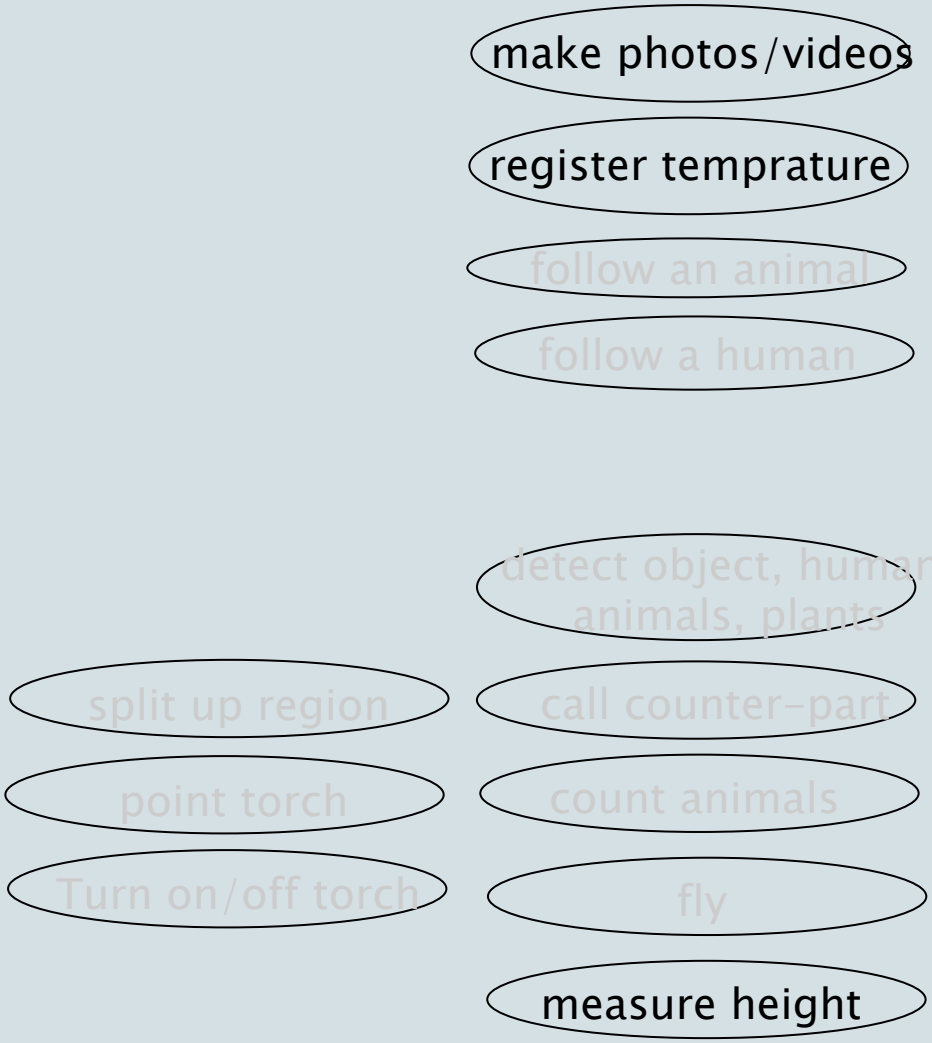


Drone SS: Health Monitor

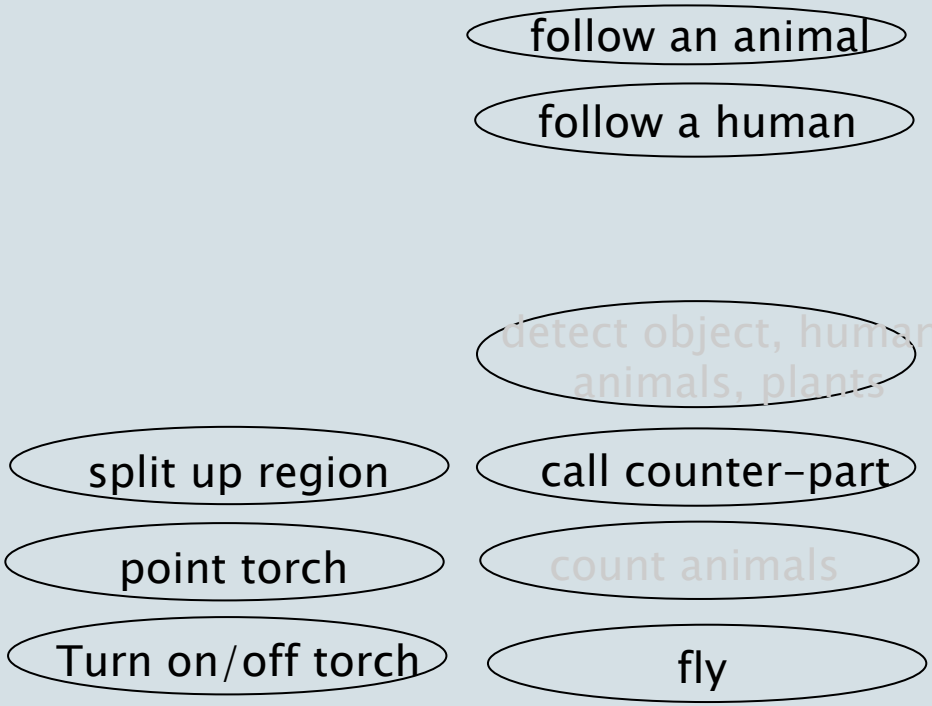


Drone SS: Communication

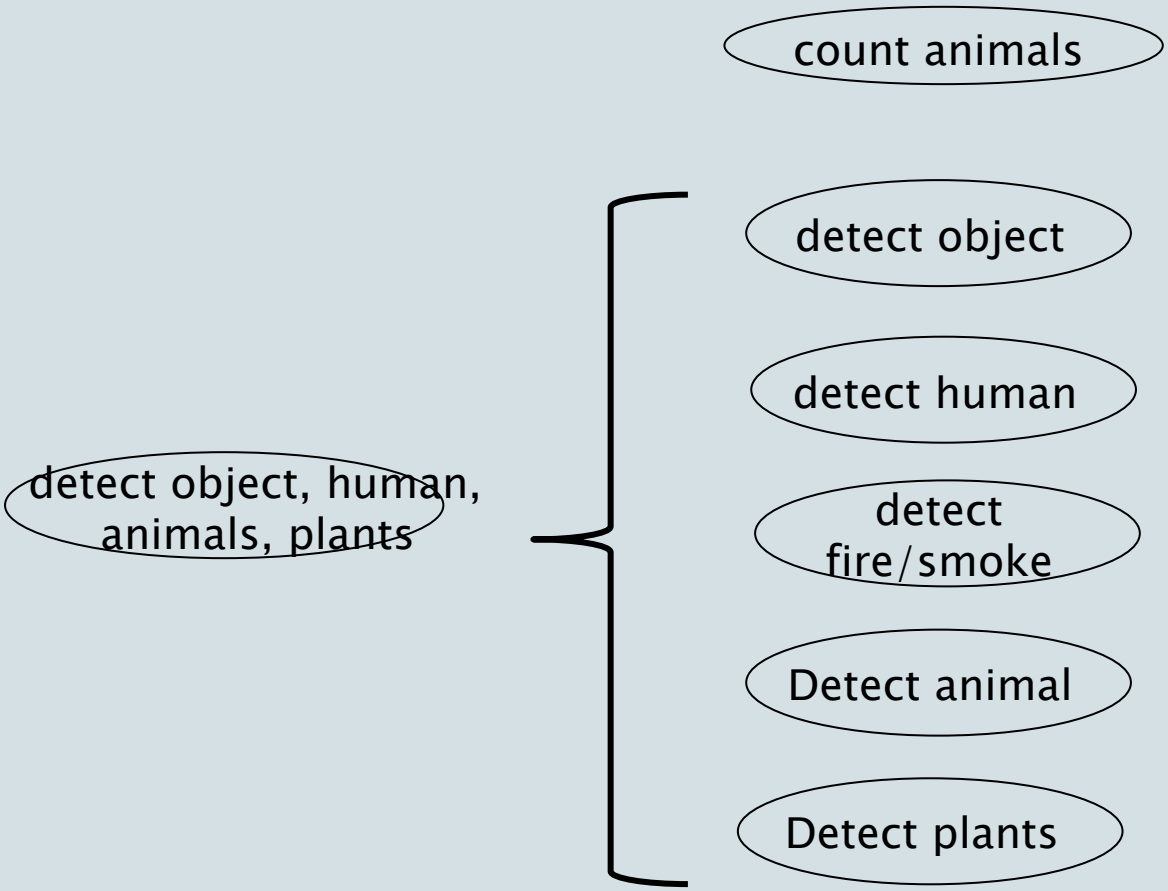
Drone SS: Surrounding Monitor



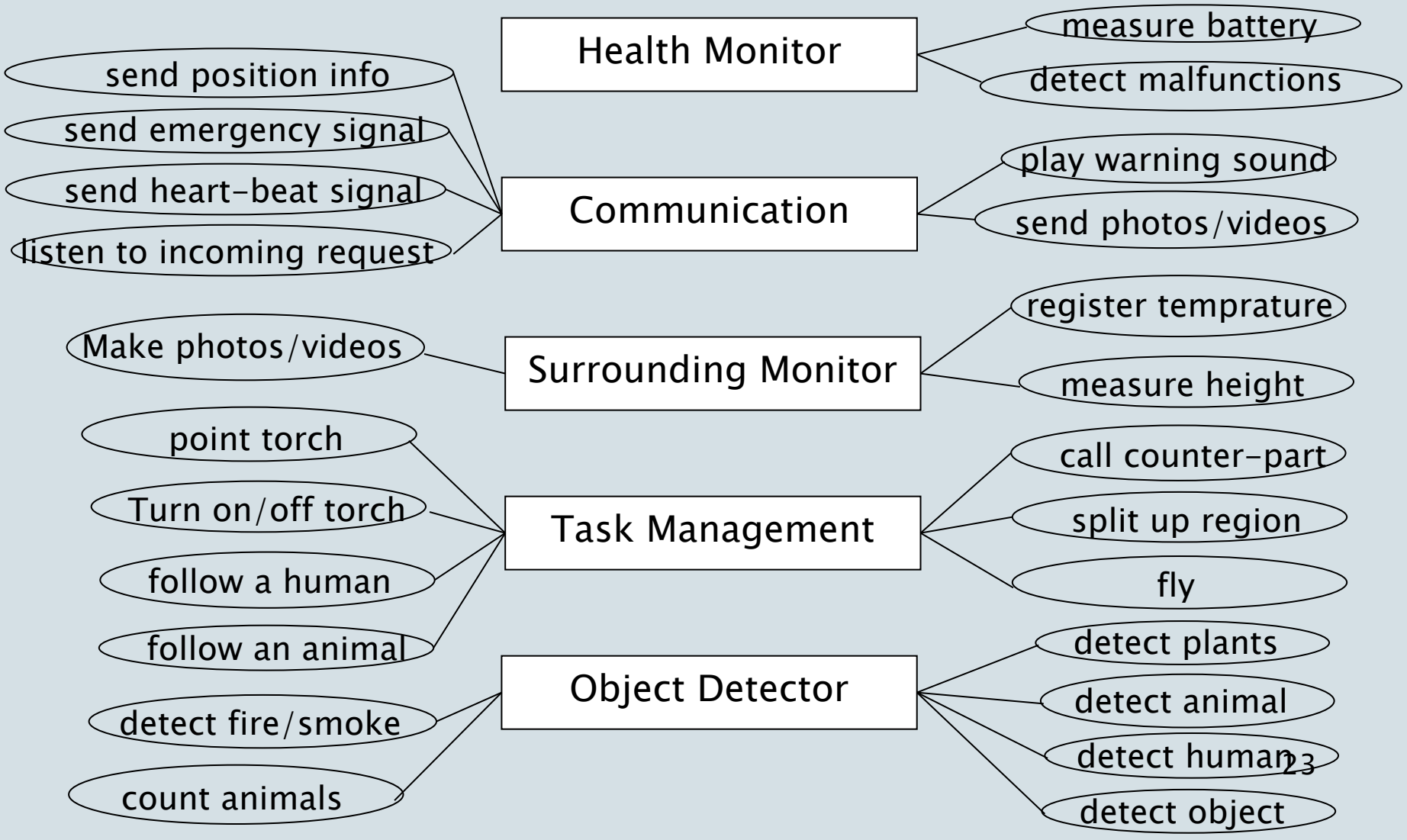
Drone SS: Task Management



Drone SS: Object Detector



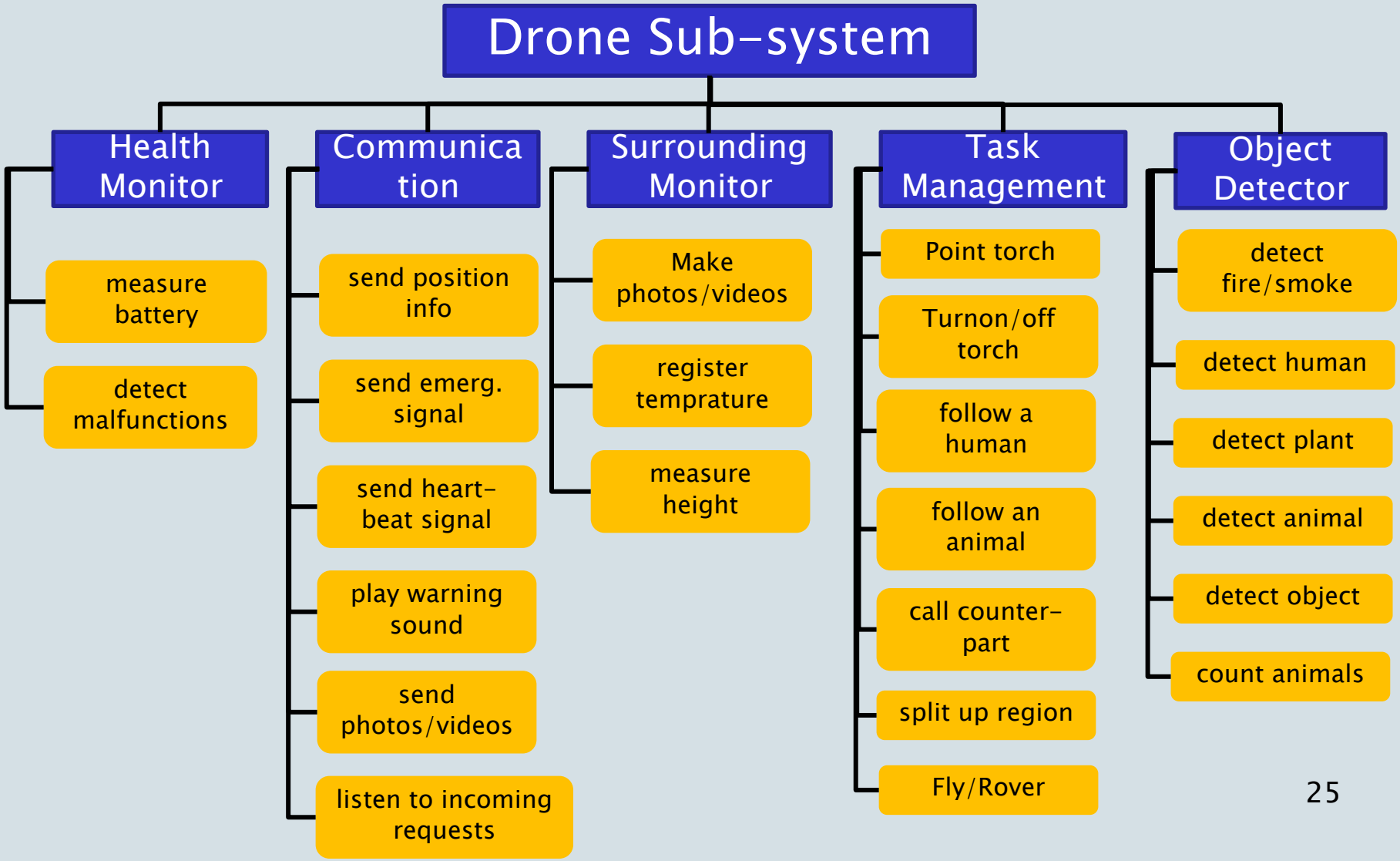
Drone Sub-system: Functionality Areas



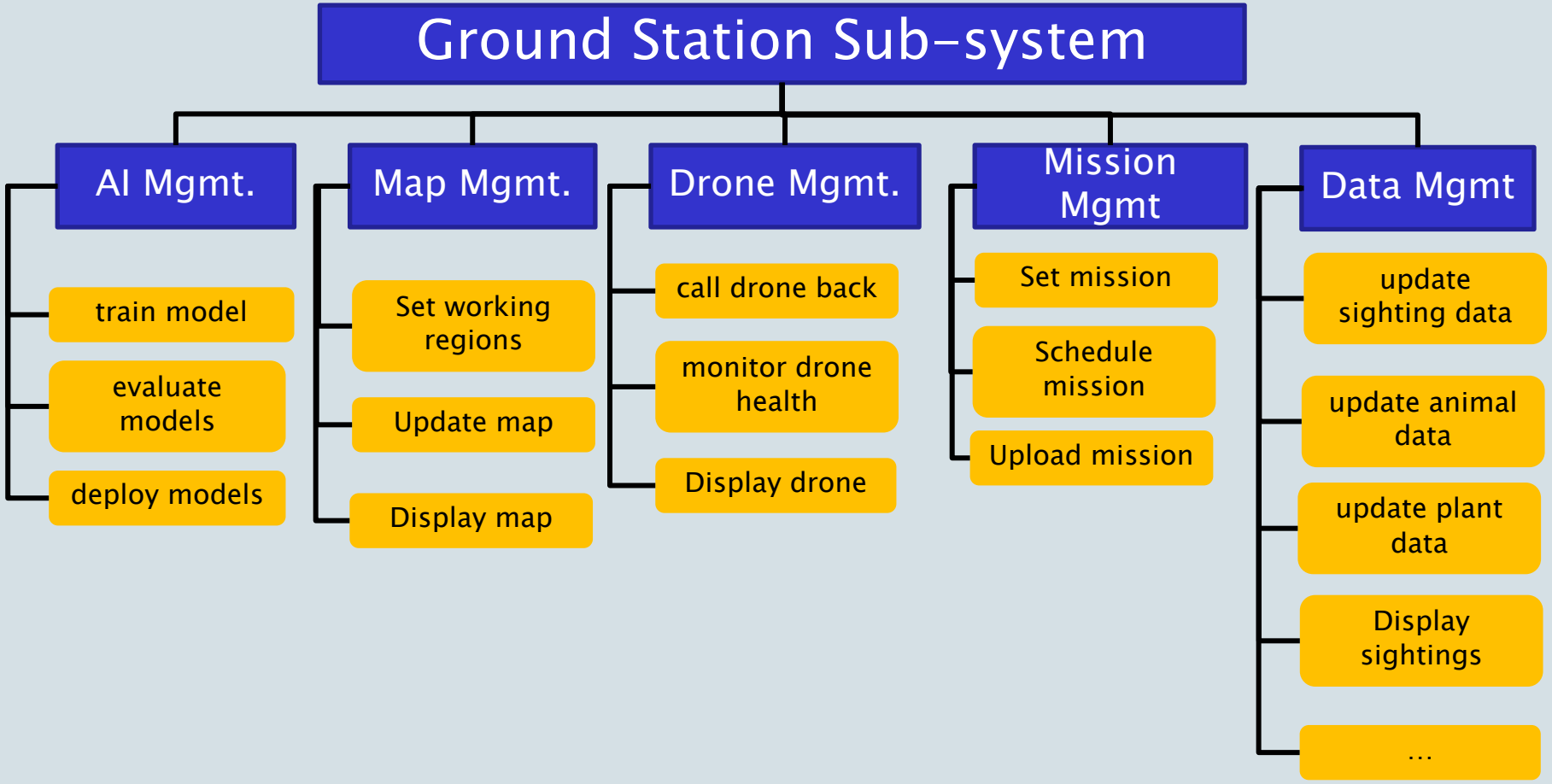
Drone Sub-system: Functionality Areas

Health Monitor	Monitor health (battery, operational time, fault) of the drone
Communication	Initiate and maintain communication between drones and other part of system
Surrounding Monitor	Monitor the condition of surrounding areas of the drones
Task Management	Manage and control drone operation. Switch between different working modes.
Object Detector	Detect and recognize objects appeared in drone's camera. (decoupled from Surrounding Monitor)

Drone Sub-system: Func. Decomposition

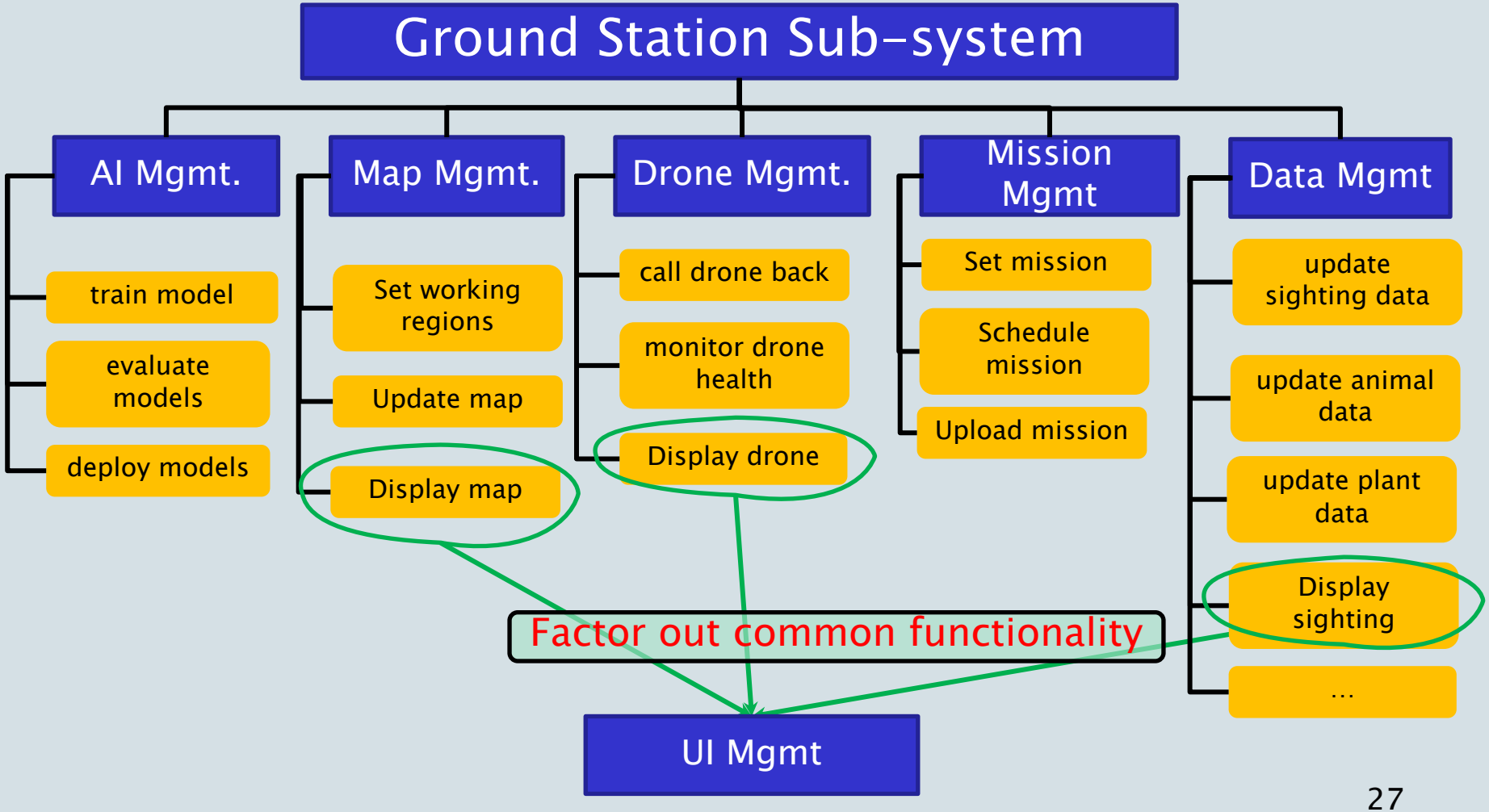


Ground Station Sub-system: Func. Decomposition (1st version)

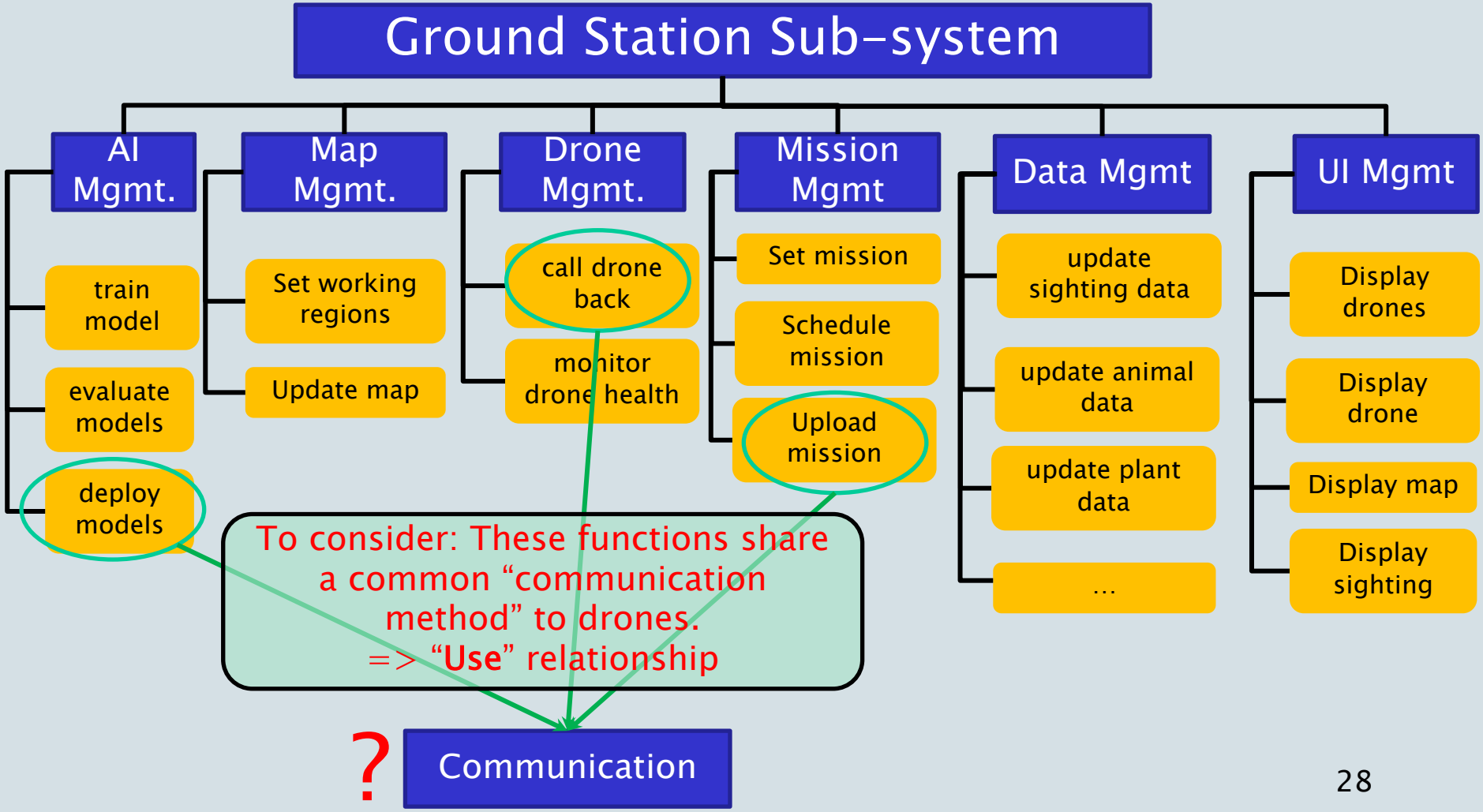


? Charging Station Mgmt

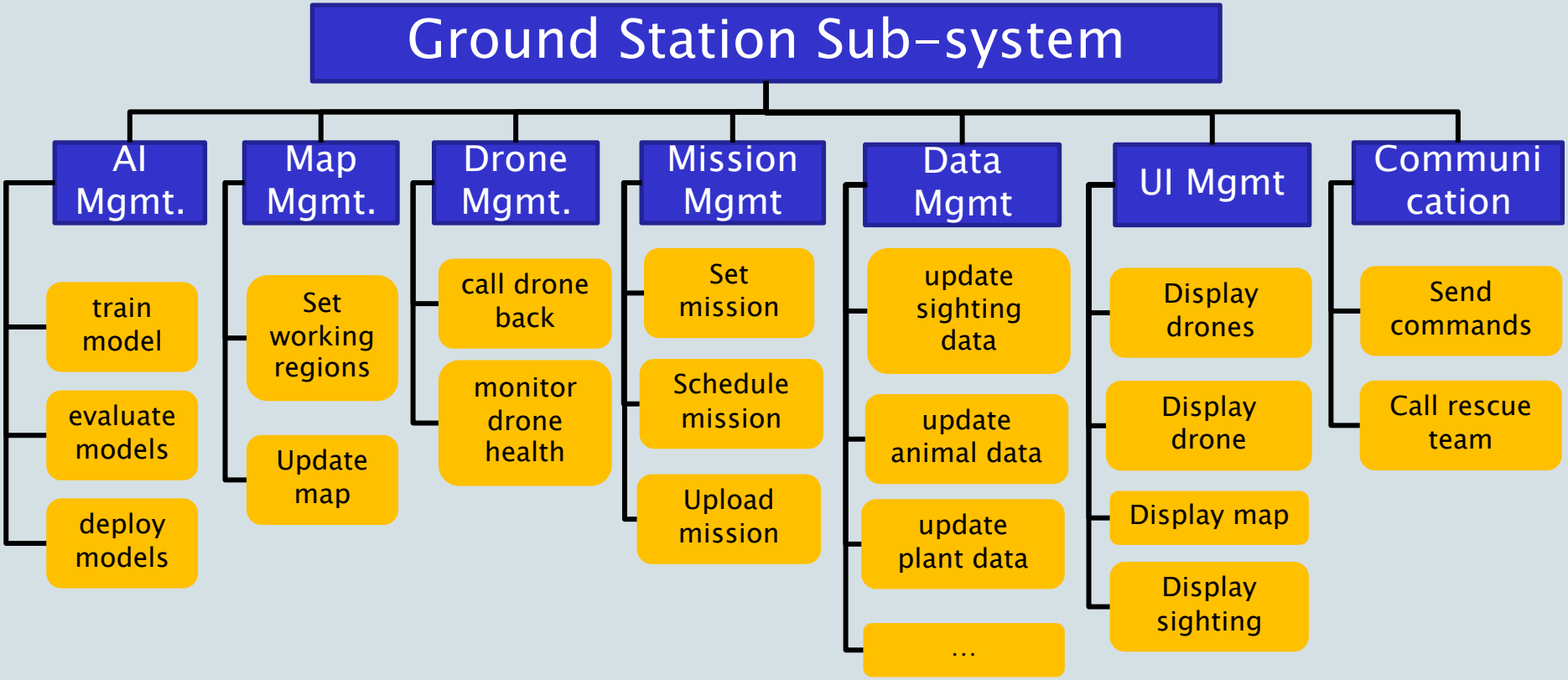
Ground Station Sub-system: Func. Decomposition (1st version)



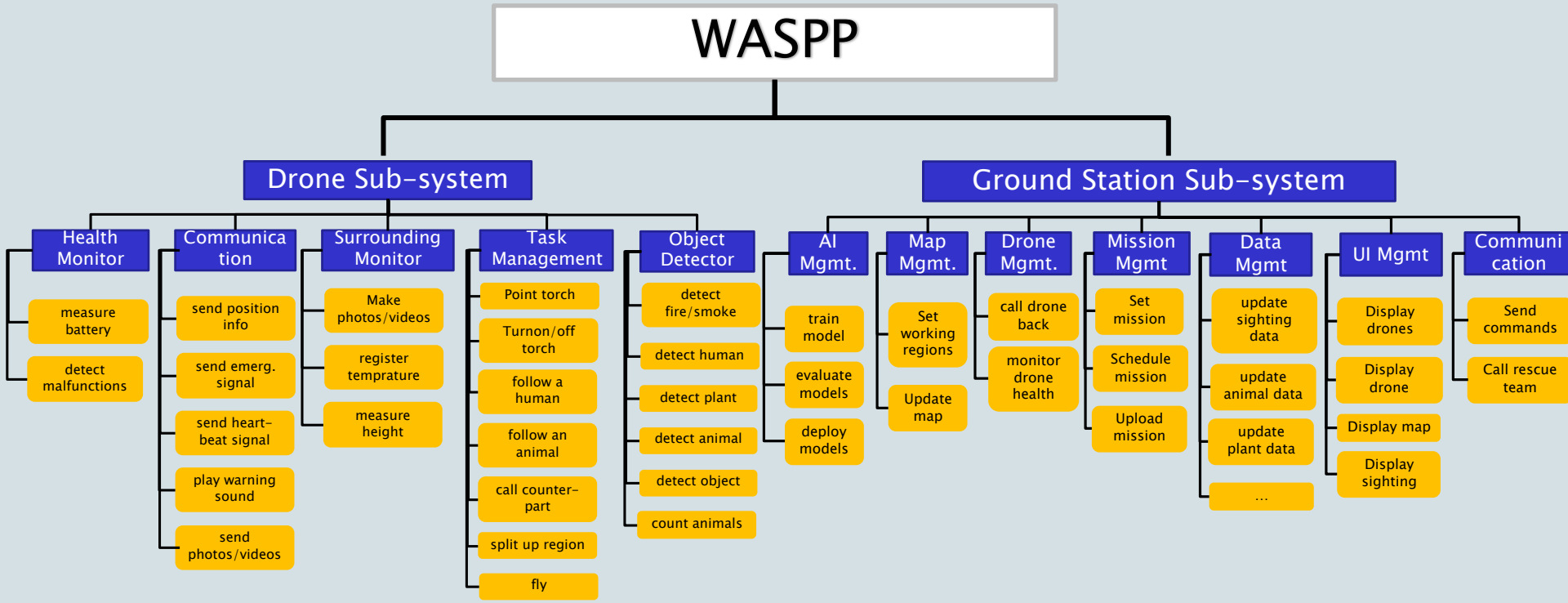
Ground Station Sub-system: Func. Decomposition (updated)



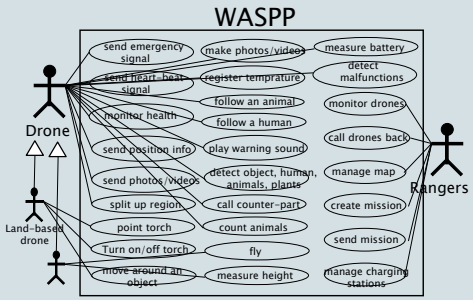
Ground Station Sub-system: Func. Decomposition (final)



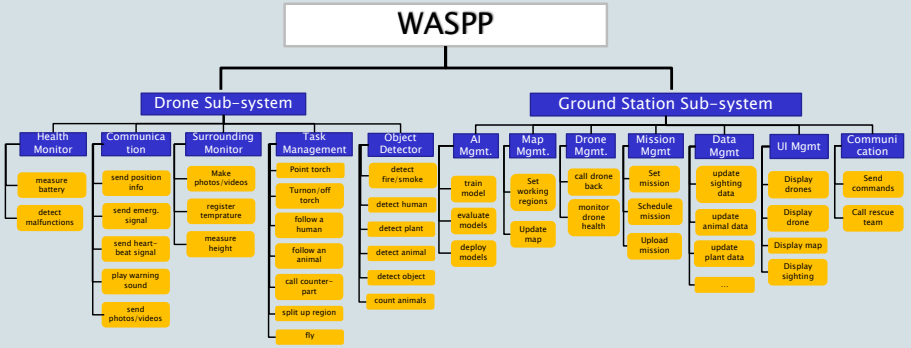
WASPP – Functional Decomposition Diagram



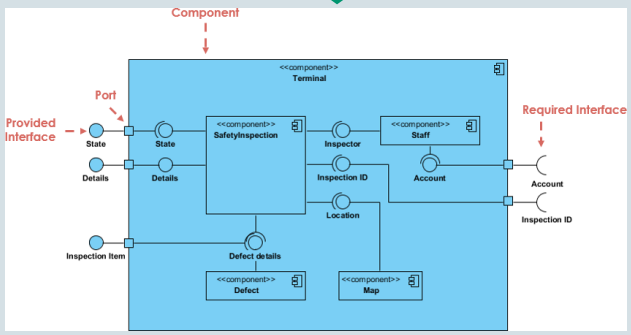
Structural View



Functionalities

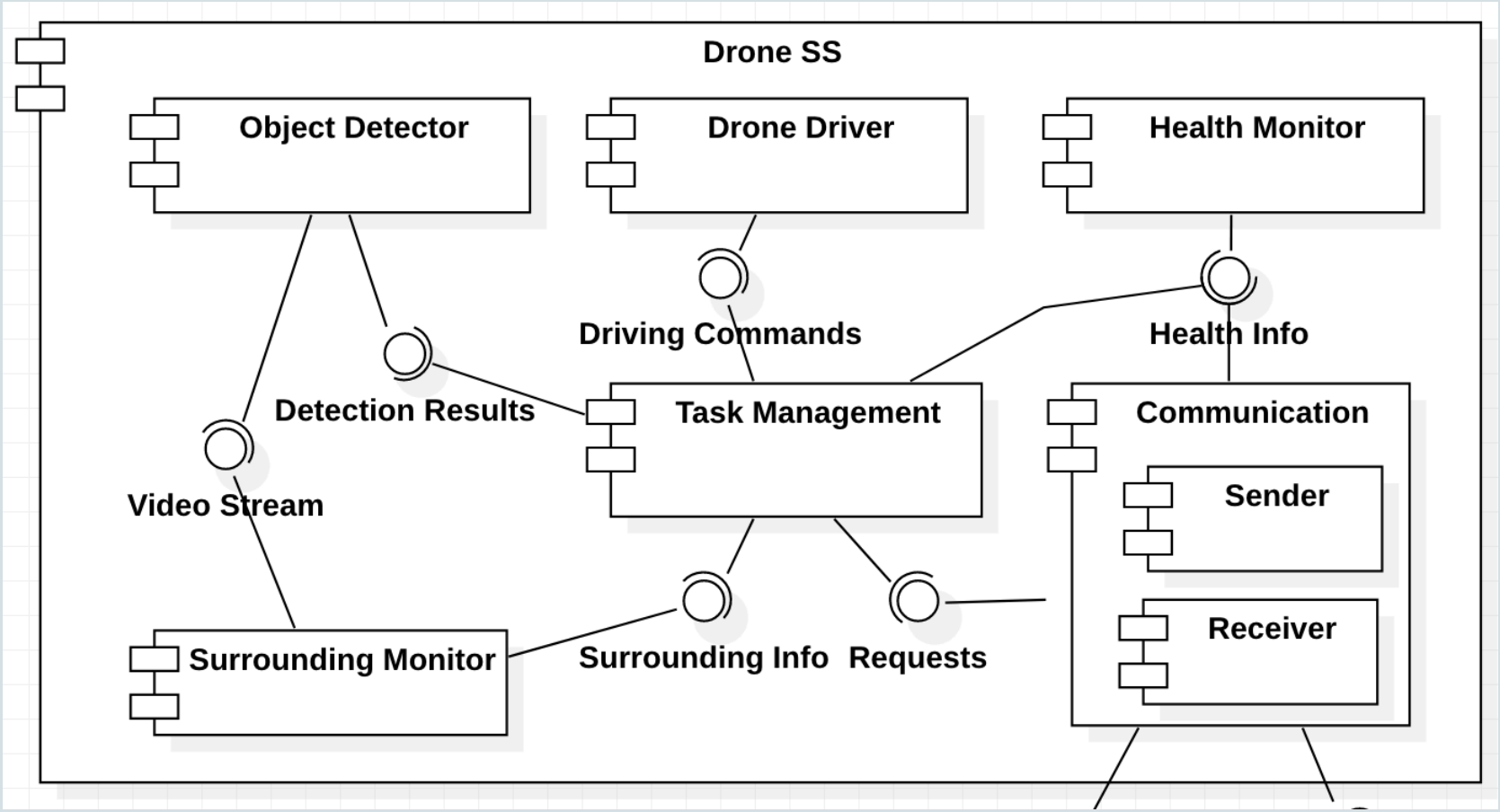


Functional Decomp.



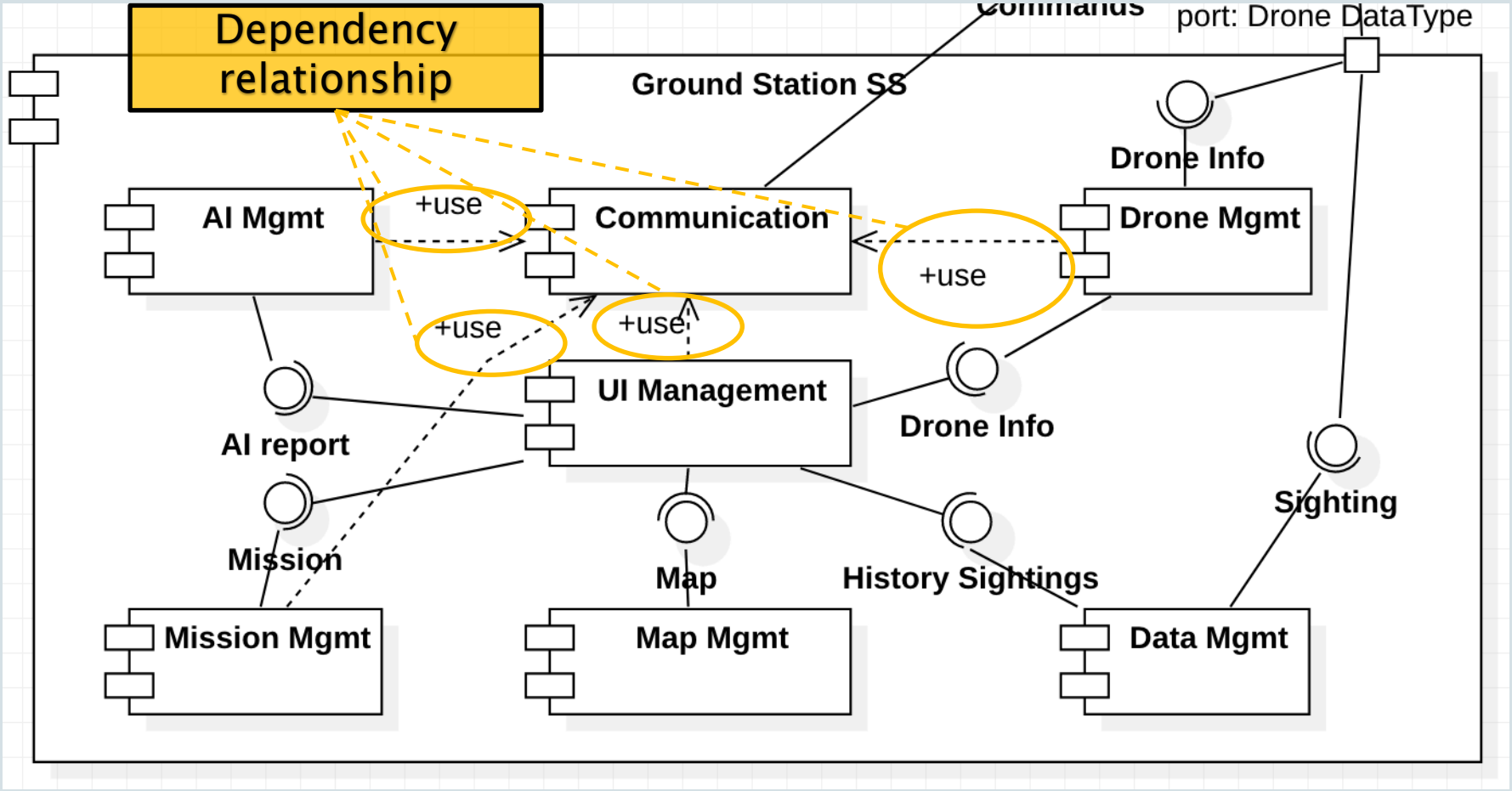
Structural View

Structural View (1)



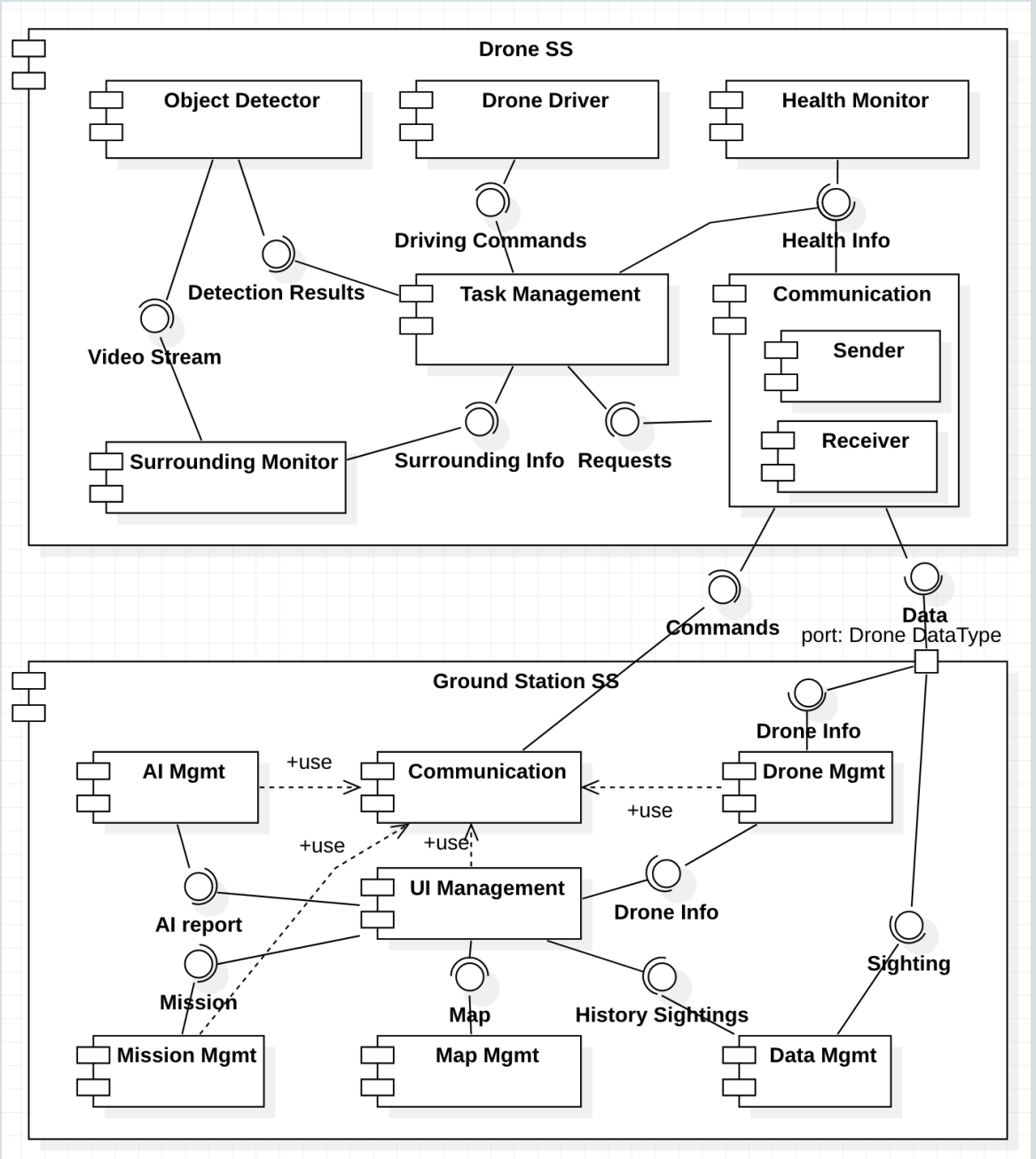
Drone SS

Structural View (2)

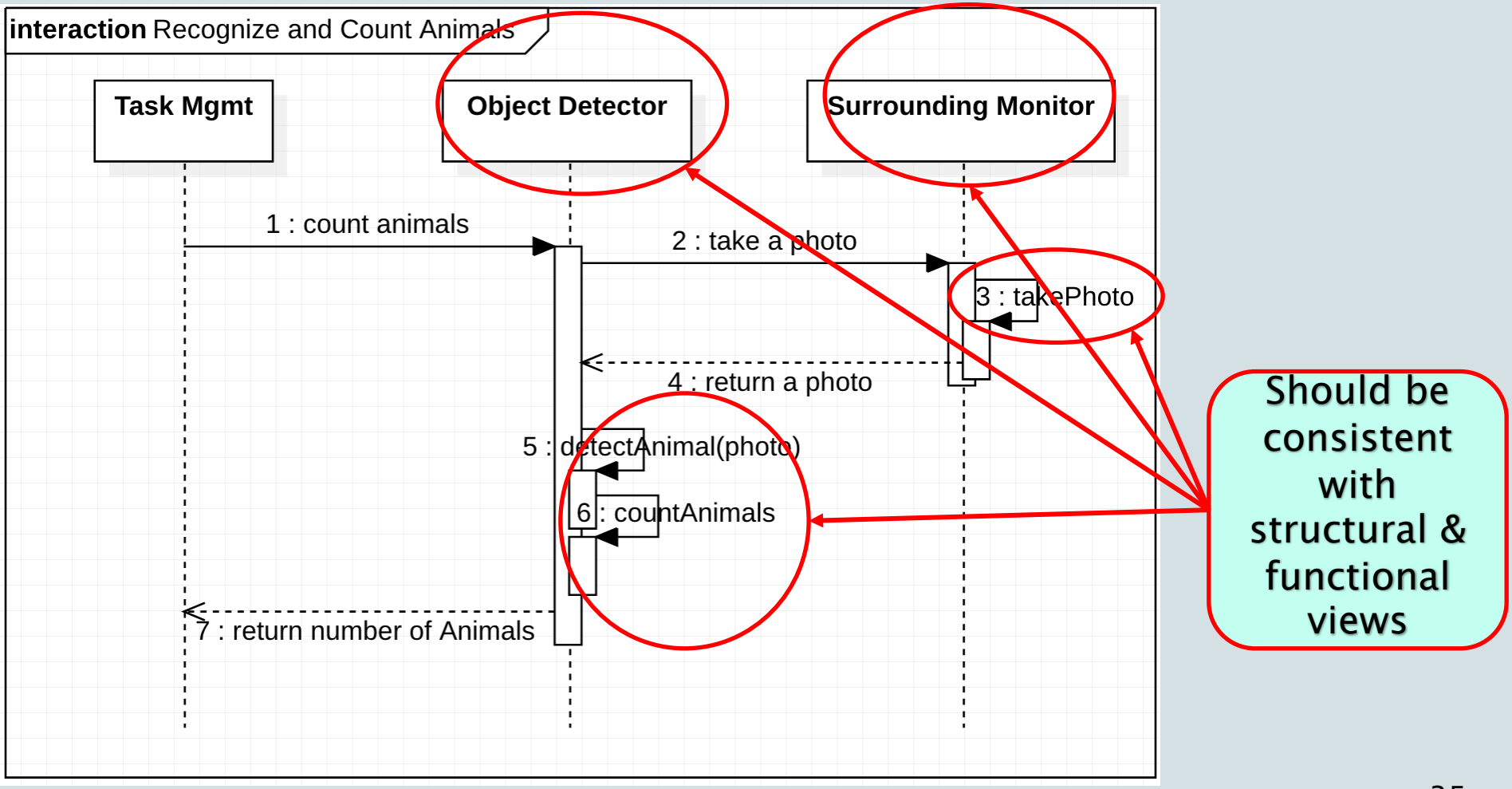


Ground Station SS

Structural View (whole system)

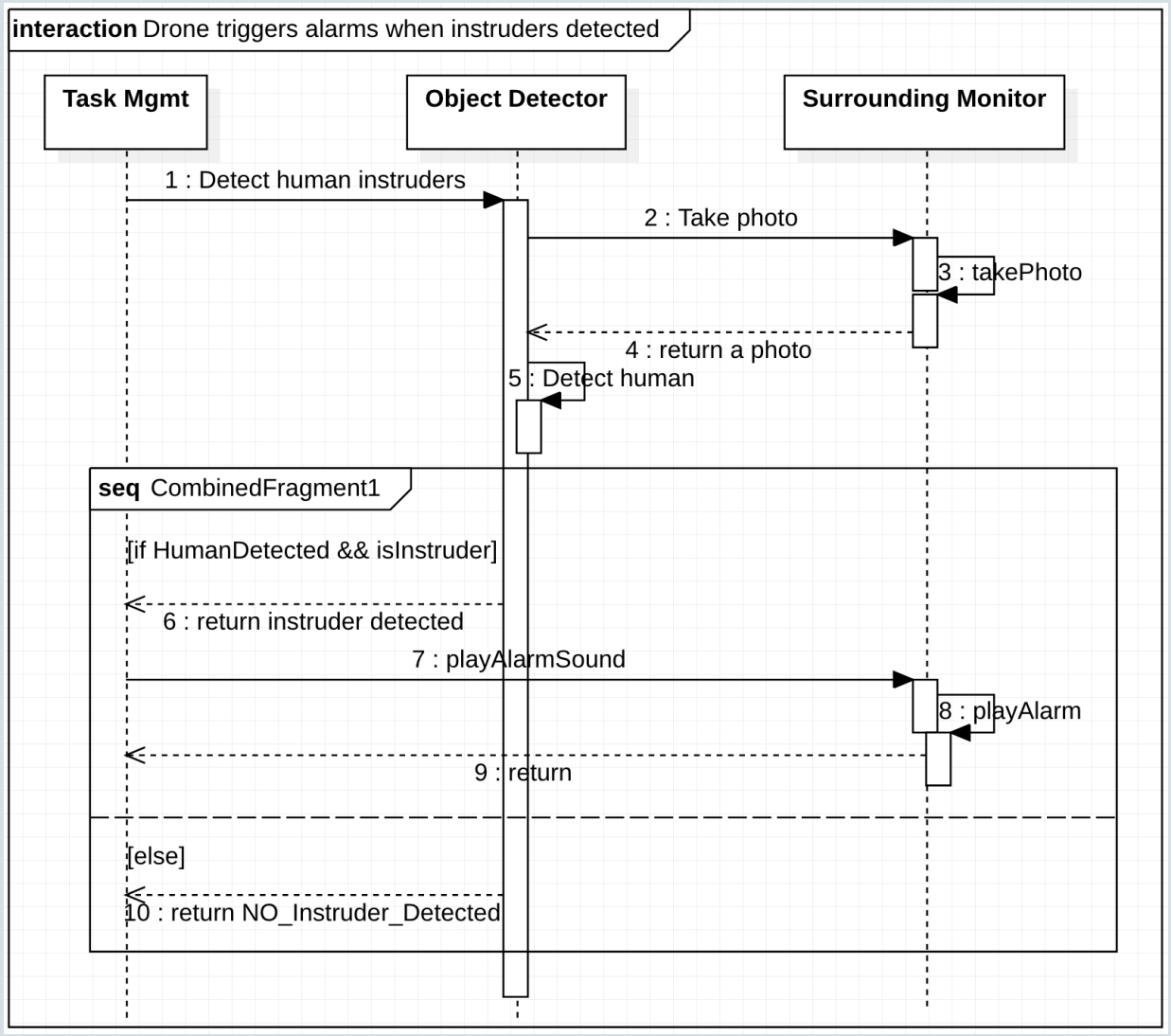


Behavioral View

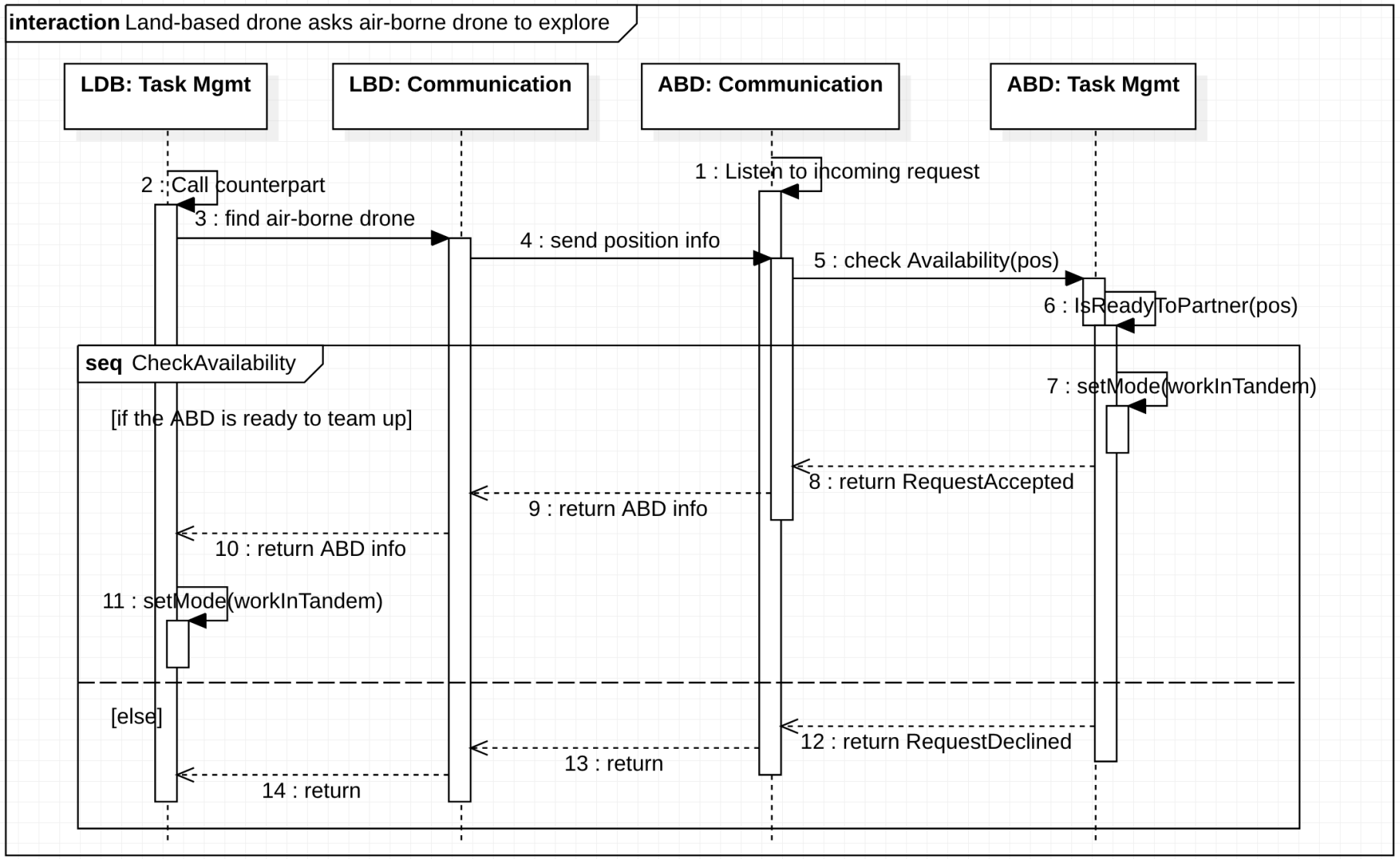


Assumption: The request of counting animals is initiated from the Task Management

Behavioral View (2)



Behavioral View (3)

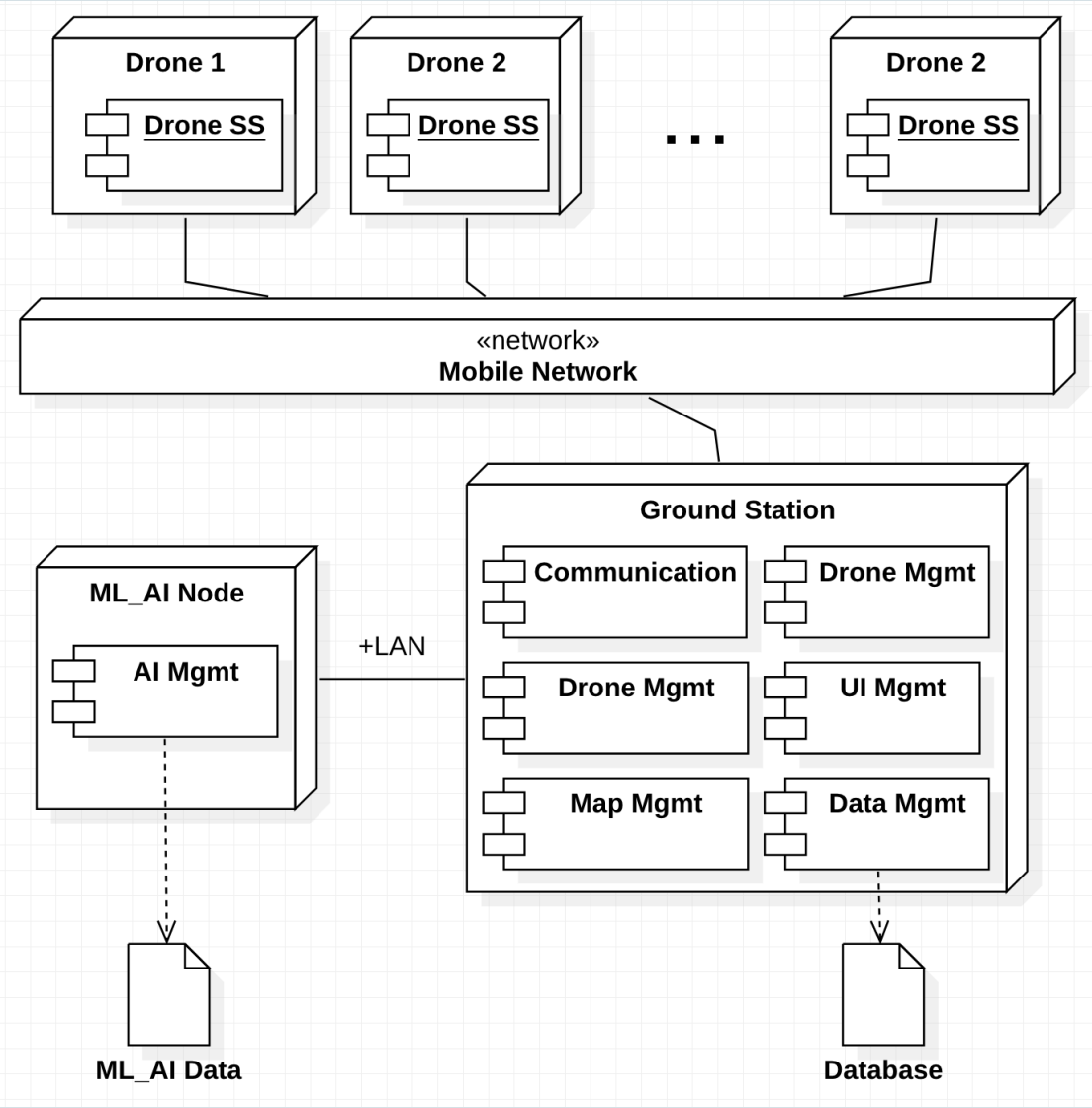


Client/Server or Publish/Subscribe

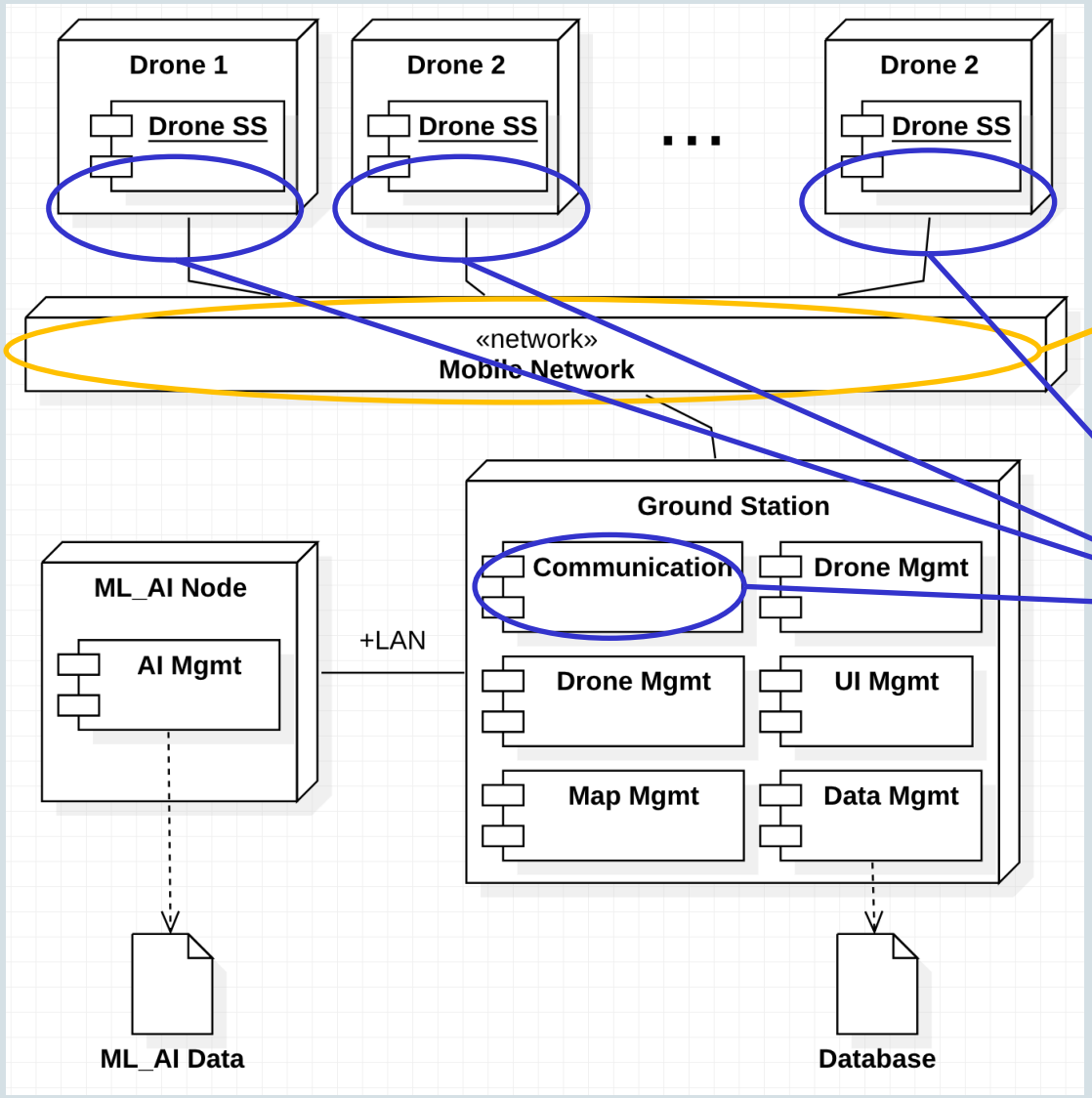
- I chose Pub/Sub as the dominant style because:
  - Short-lived data
  - Multiple receiver (subscribers) of “specific” datatype
    - Example: A drone is interested in “position” but not “image/video stream” of other drones
  - Flexibility when adding/removing subscribers or publishers.

- I do not choose Client/Server because
  - Frequent communication between drones
  - Some requests should not primarily be made by drones
    - E.g. I don’t want drones (represent client-side) to send requests every N seconds to see if there is any update on the mission from the ground station (represents server-side).

Deployment View



Deployment View



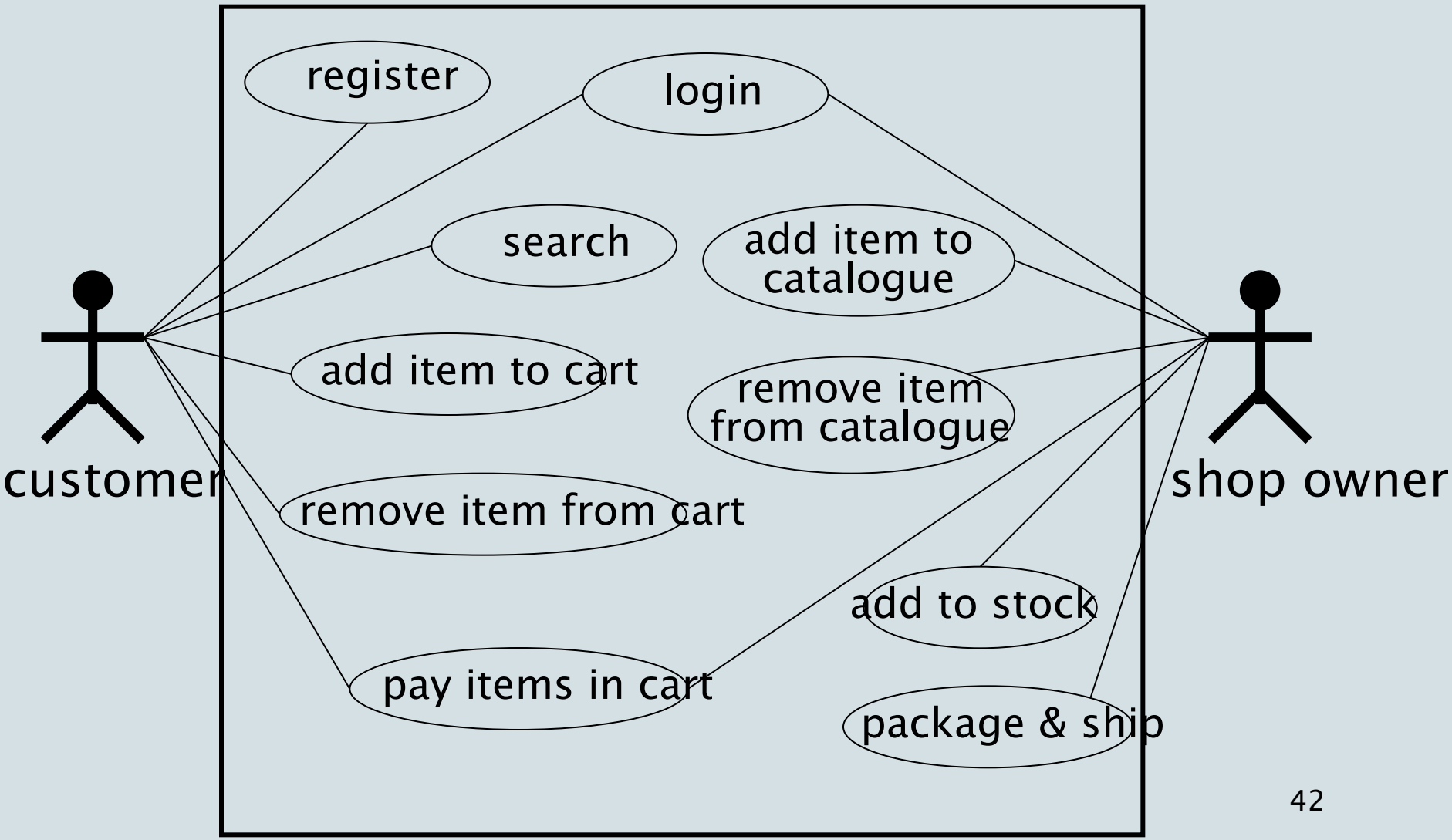
Pub/sub middleware

Communication Component can cover the role of a Pub/Sub network Agent

and… More to think of

- Don’t forget architecture drivers
  - You should be able to answer: how were your design decisions driven by the drivers?
  - Where?
    - Chosen styles
    - Tactics
    - Additional components, deployment nodes …
  - Use your SAD to document the decisions
- Level of details
  - Think of: Who are the readers? To do what?
- Consistency
  - Any adjudegement to be made to requirements?
  - Keep your design (via the views) consistent

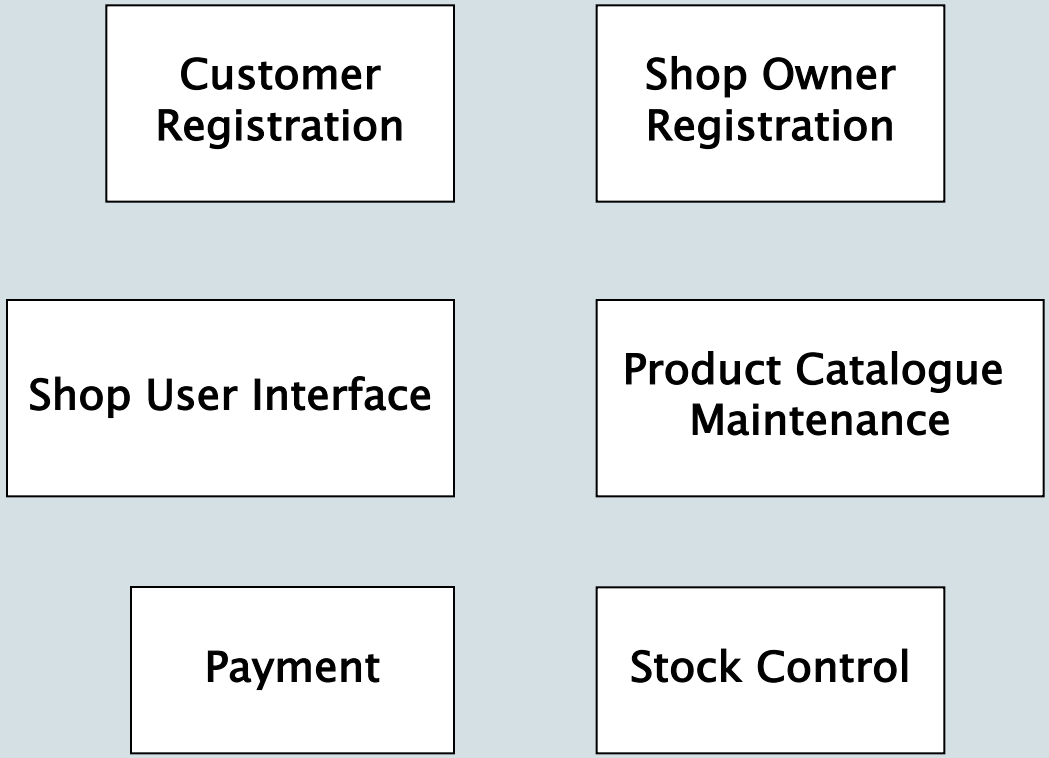
Case: Web shop



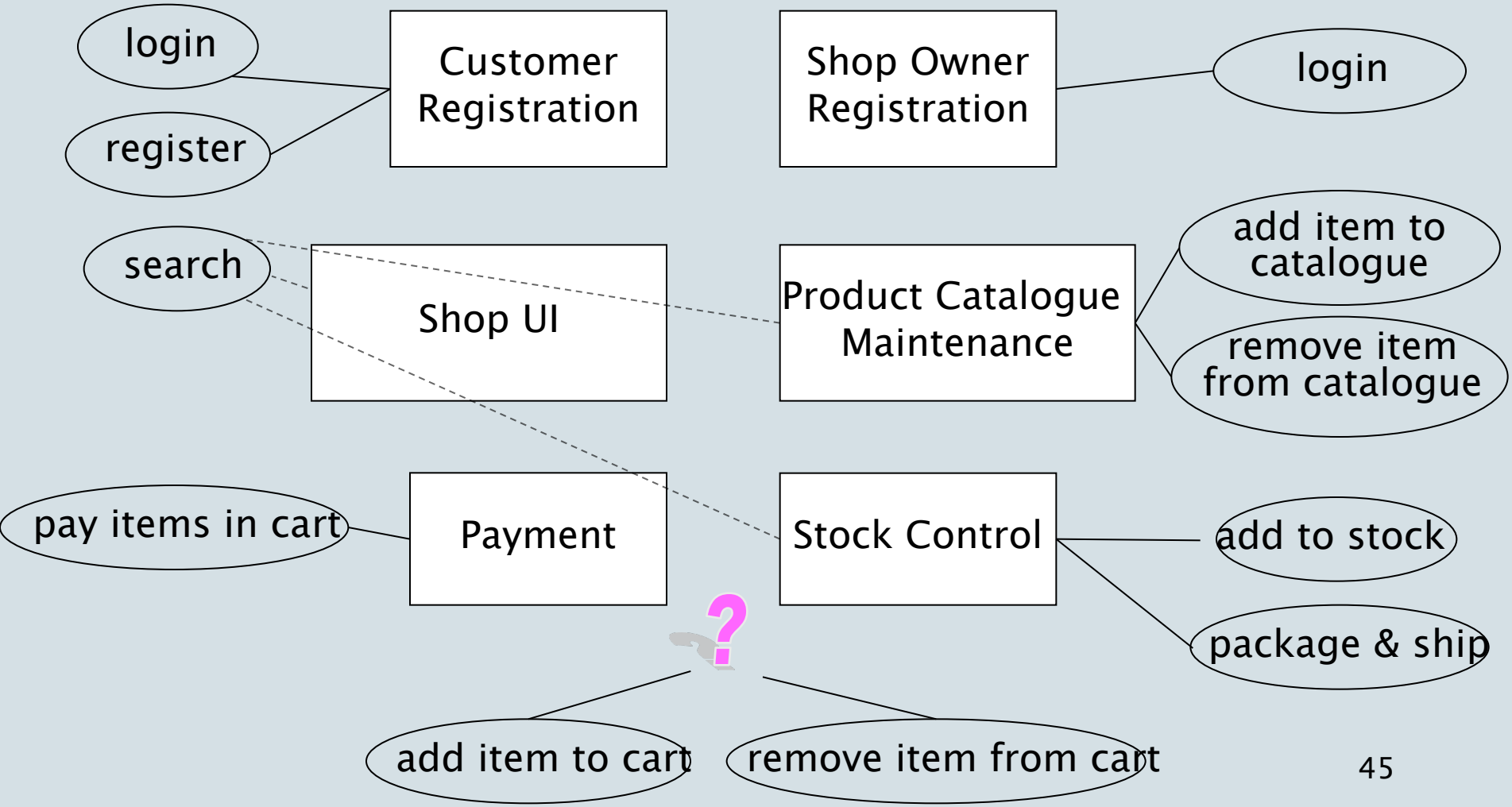
Structure Diagram

- Defines subsystems of functionality
- Purpose
  - Define decomposition into subsystems
  - Provide support for use-cases
- Use Component (or Class) diagram

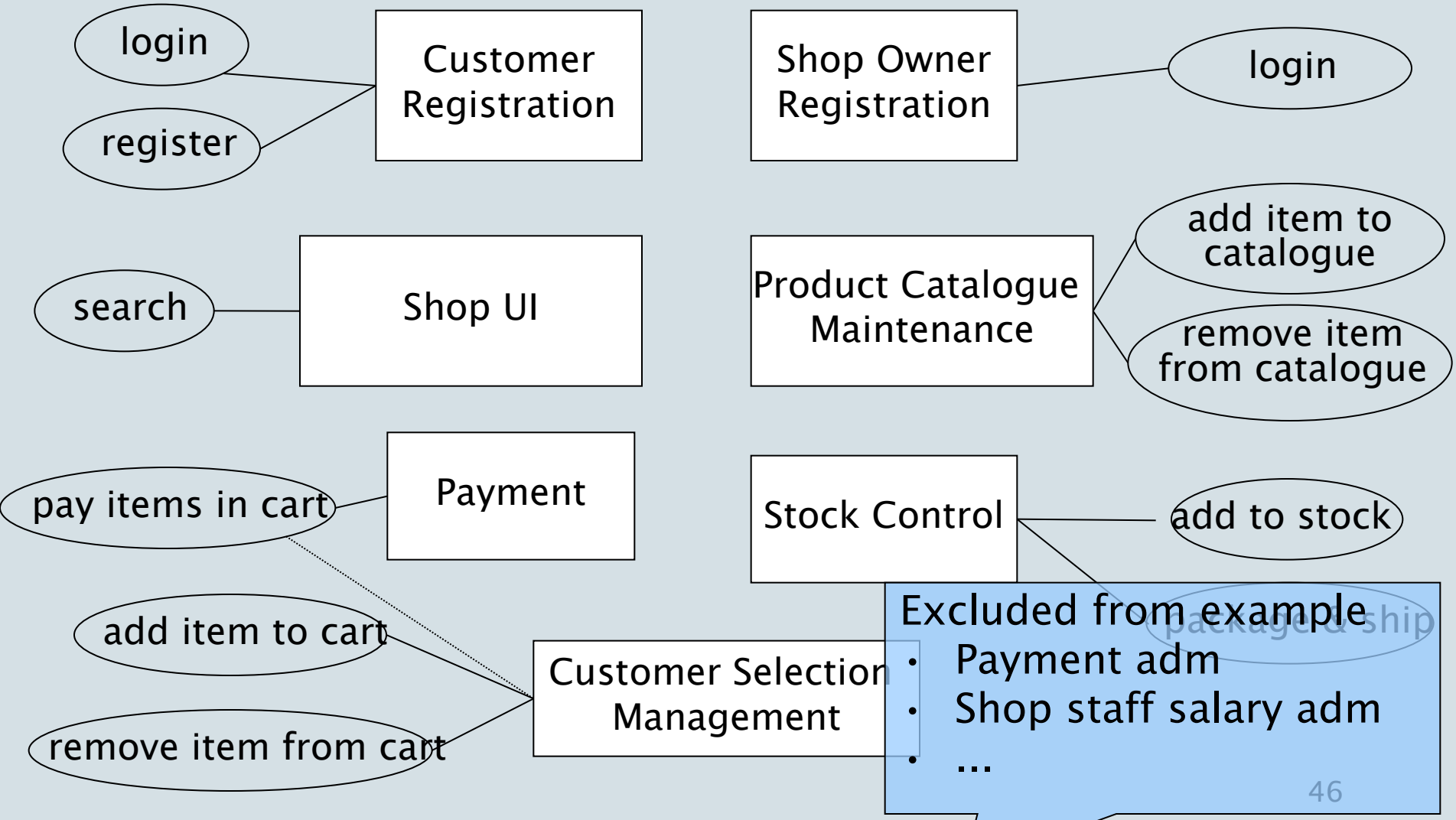
Web Shop: Functional Areas (V0.1)



Check Use Cases Against Functional Areas



Web Shop: Functional Areas (V0.2)



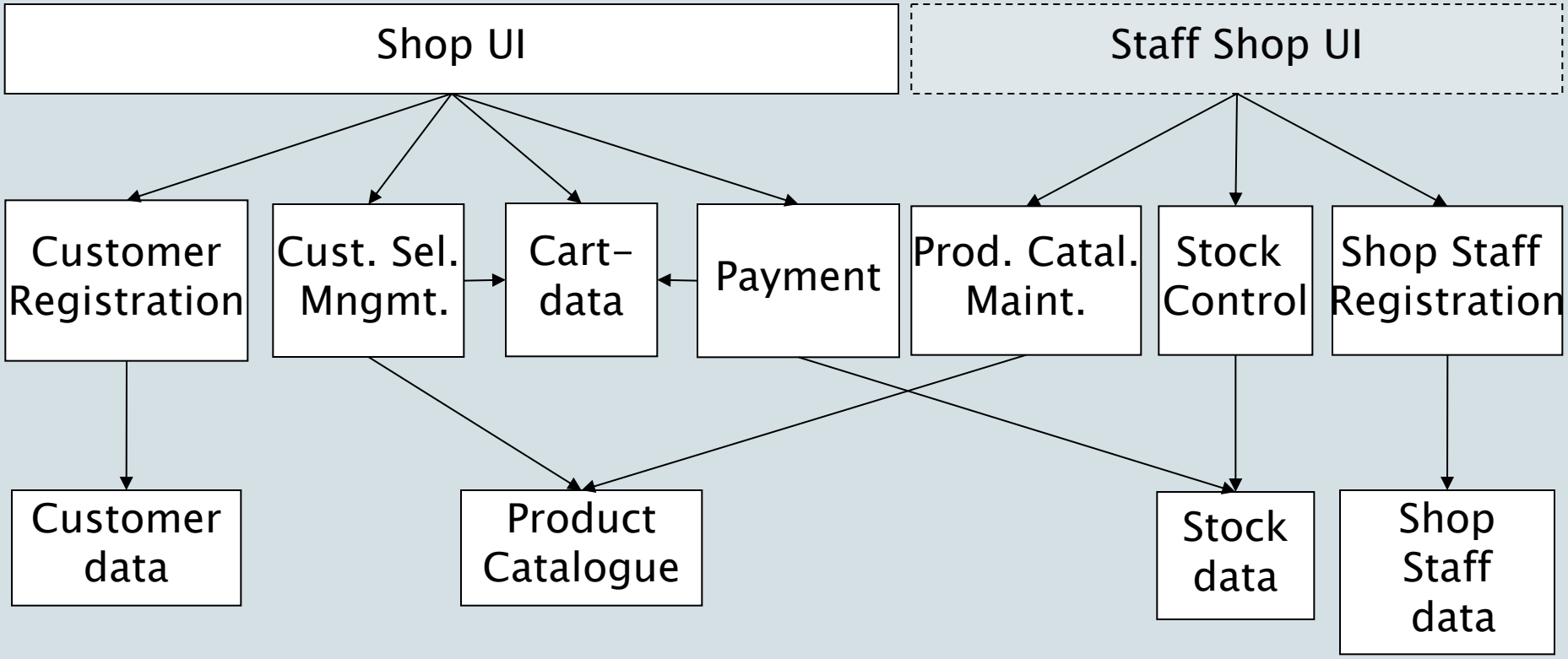
Web Shop: Responsabilities

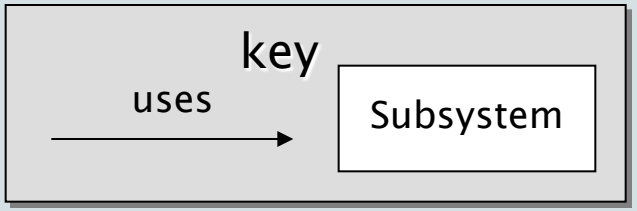
Customer Registration	Entry, storage & retrieval of customers
Shop Owner Registration	Entry, storage & retrieval of shop staff
Shop UI	Provide customers access to product data
Prod. Cat. Maintenance	Entry, storage & retrieval of product data
Cust. Selection Mngmt.	Register customer product selection
Payment	Handle transaction between customer & shop
Stock Control	Register available products in stock

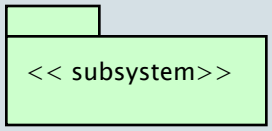
Some Concrete Examples

- http://www.nkictarchitectuur.nl/2006/index.htm
  - Rabobank, Belastingdienst, Ahold, FEI, Schiphol, Elsevier, …

Identification of Dependencies

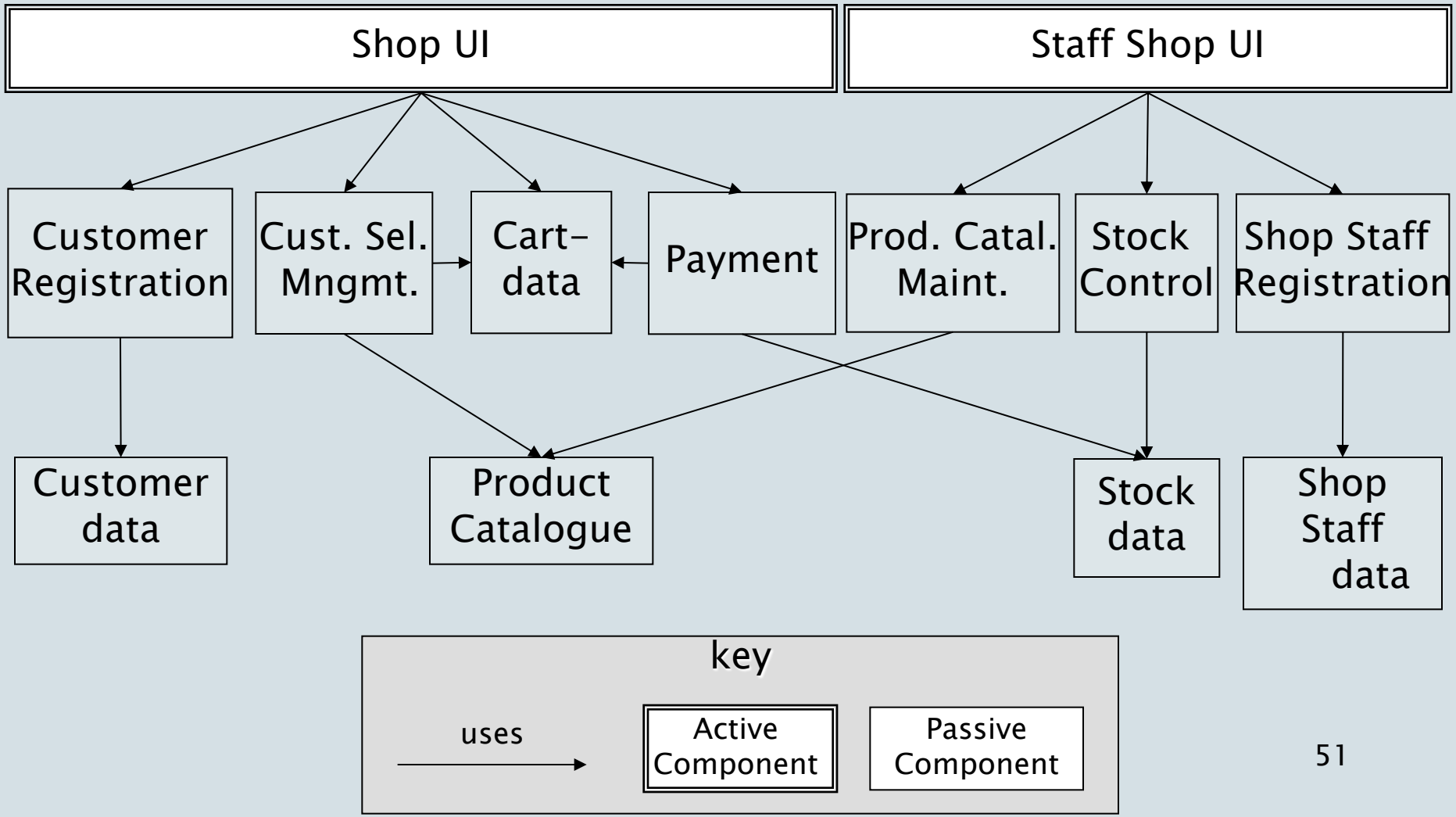






package

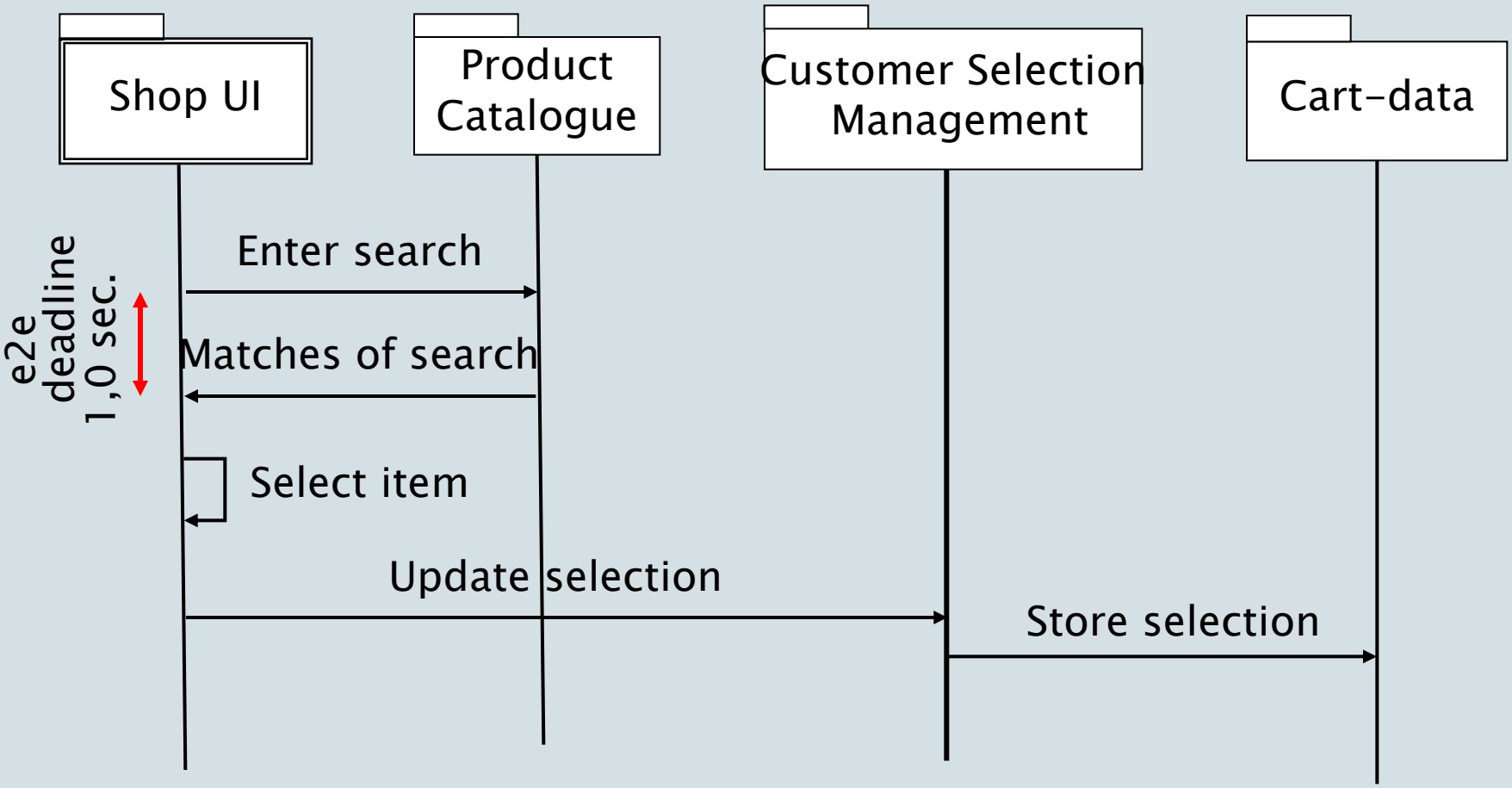
Identification of Active Processes



Sequence Diagram

- Captures interaction between components
- Purpose
  - Model flow of control
  - Identify synchronization
  - Illustrate typical scenarios

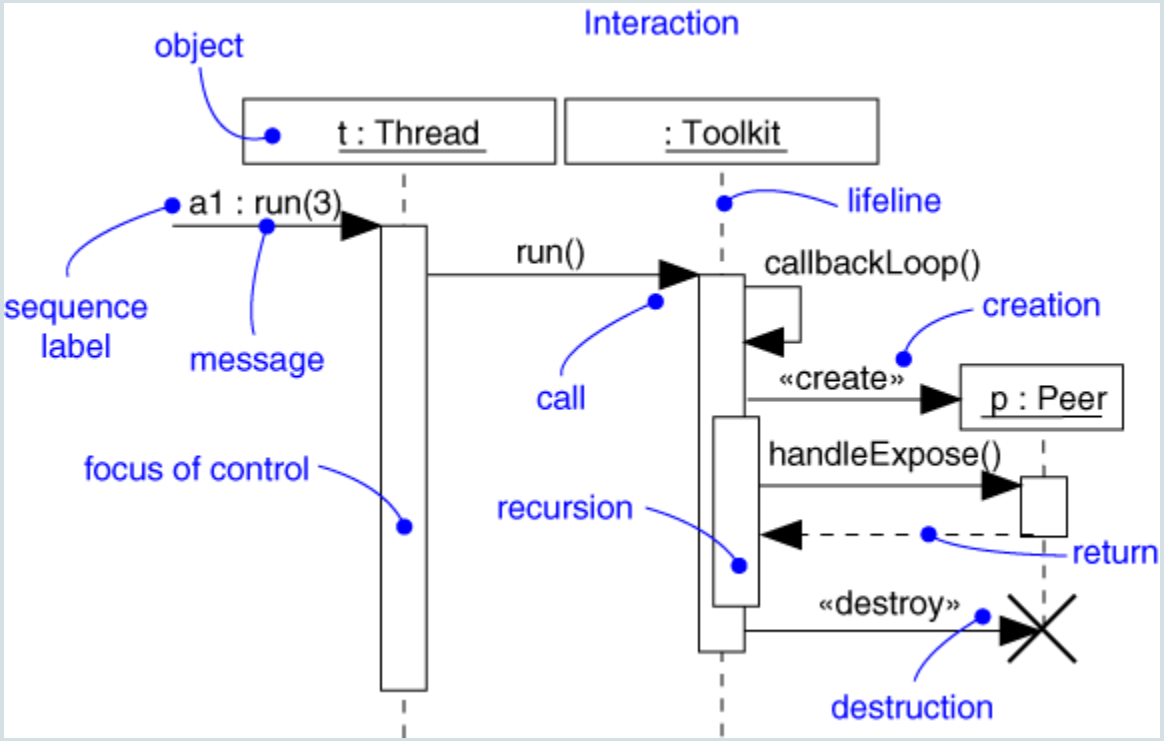
Design System Dynamics: Scenario's



Scenario's describe the interaction between components

Sequence Diagram

Captures interaction between components

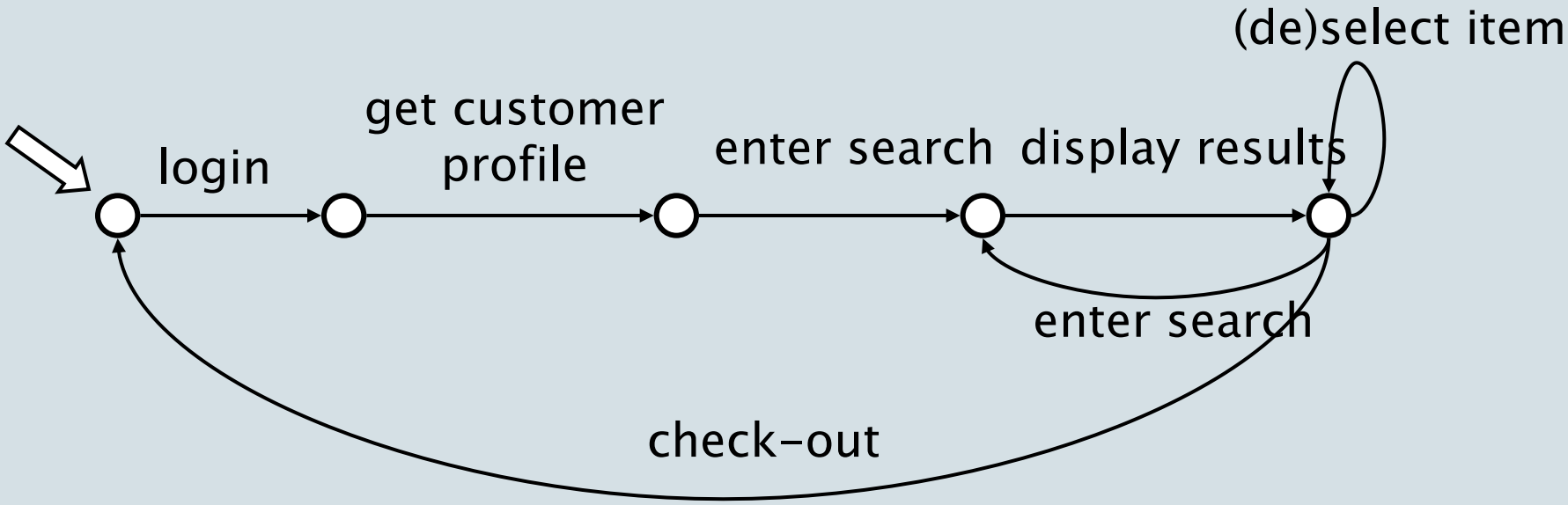


Statechart Diagram

- Captures behaviour internal to individual components
- Purpose
  - Model reactive objects
    (user interfaces, devices, etc.)

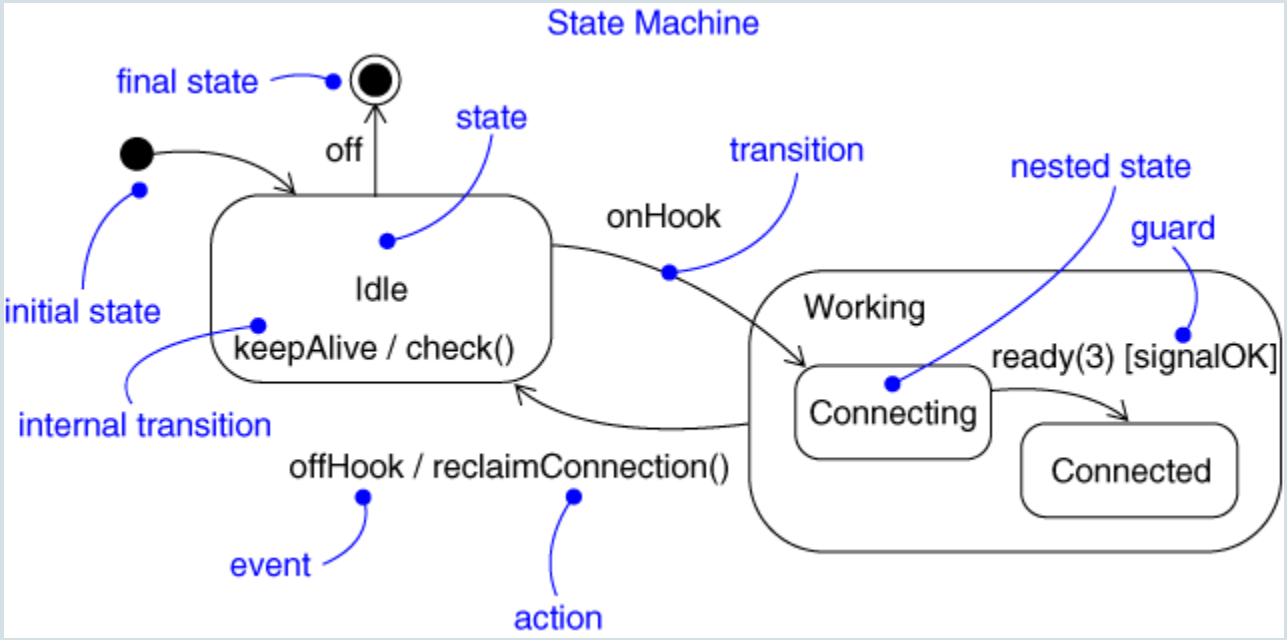
a.k.a. state-transition diagrams, finite-state-machines
alternative: Petri-nets

Web shop UI state-machine



Statechart Diagram

Use a state-diagram to specify the internal behaviour of a component

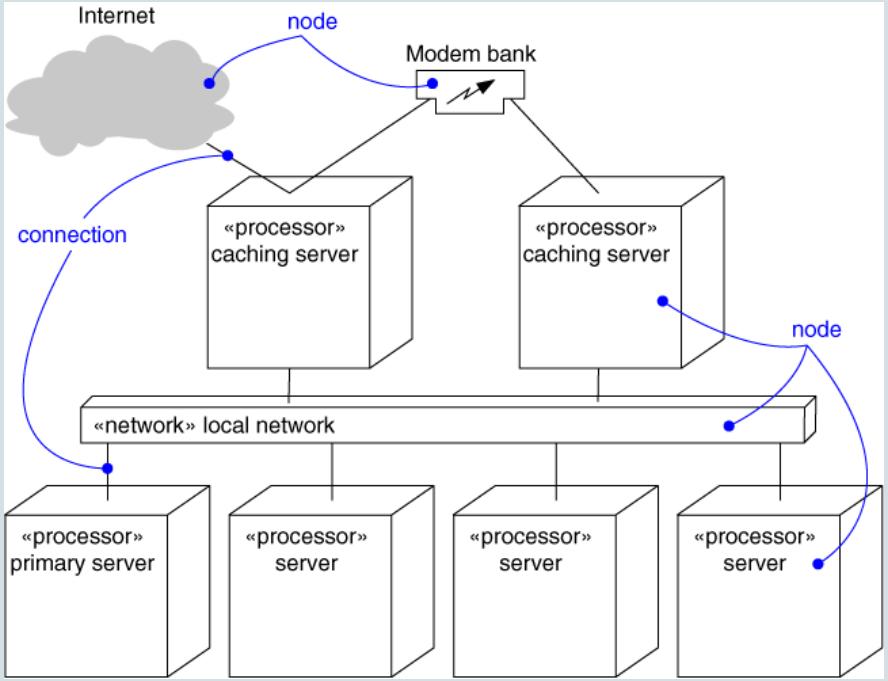


Deployment Diagram

- Captures the topology of a system's hardware
- Purpose
  - Specify the distribution of components
  - Identify performance bottlenecks
- Developed by architects, networking engineers, and system engineers

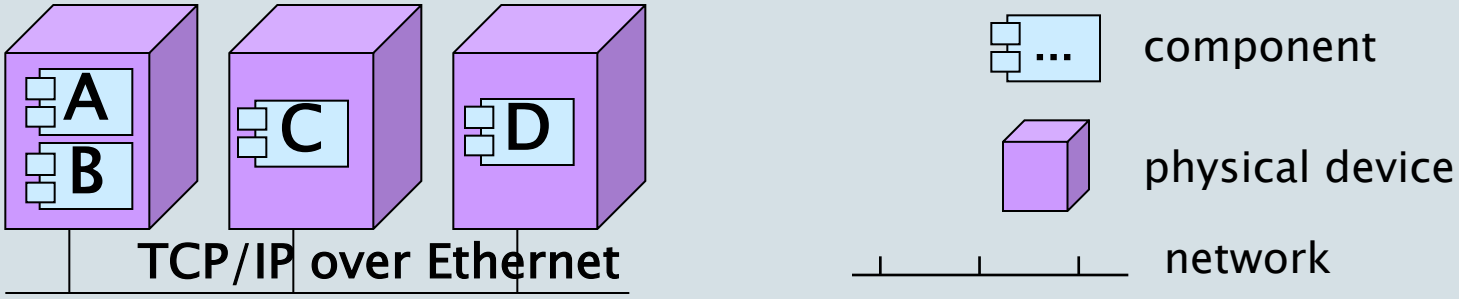
Deployment Diagram

- topology of a system's hardware
- + (rules for) mapping of logical view to hardware



Deployment Diagram

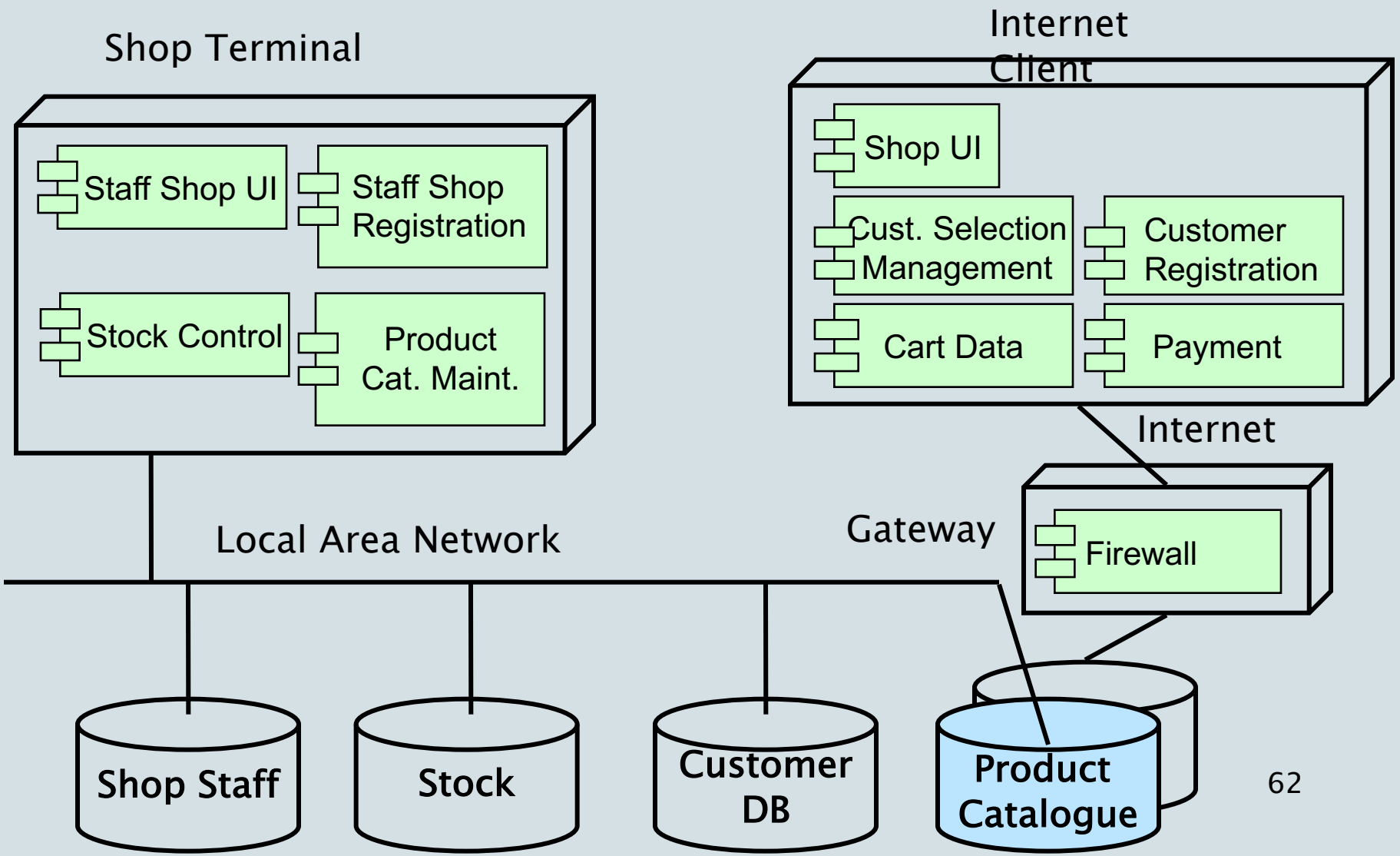
Deployment view: physical model + mapping



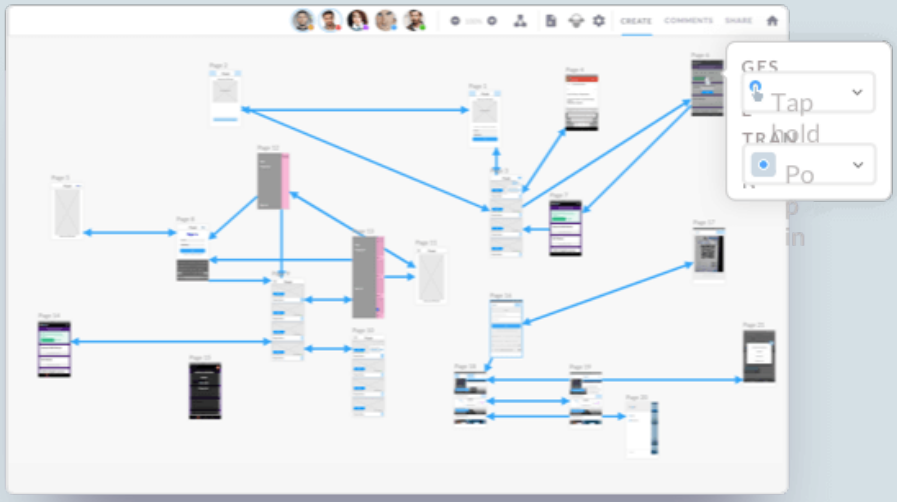
Basis for analyzing throughput, availability

Separation of Concerns in diagrams: the deployment diagram does not show the dependencies between components

Web Shop Deployment Diagram



Task 2 – Implementation



MQTT

Node-RED

mosquitto

paho

Task 2 – Objectives

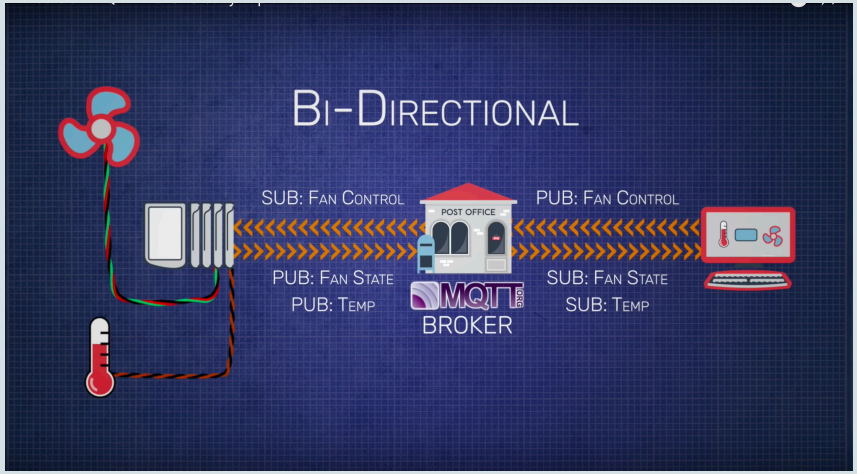
- To learn to use the architecture design as part of the implementation
- To learn to monitor that the implementation follows the architecture
- To keep the architecture description and the implementation consistent (‘in sych’) – while sometimes insight from the implementation require updates to the architecture description.
- To get practical experience in implementing architectural styles and patterns

Task 2 – Tasks

- Available on Canvas.
- To build “mockups” of main components in your system
- Mockups <> Prototype | Balsamiq
  - Show communications
  - Show behaviours, not algorithms
- Some important components:
  - Human
  - Drone (generates signals, location; receives missions…)
  - Dashboard (at Ground Station)
  - Machine Learning

Task 2 – Implementation

- Languages/Framework of your choice
- For communications between nodes, we recommend you to use MQTT
  - a lightweight (pub/sub) messaging protocol
  - avoid direct connection between devices
  - commonly used in many IoT applications



Task 2 – Starting points

- About MQTT
  - Internet
  - Introduction presentations
- Think of implementation details
  - Data structures
  - Interfaces (API)
  - Performance bottle-necks

Task 2 – Expected outcomes

- Final version of your SAD document (first version was submitted in A1T1). I expect updates:
  - based on the feedback you have from teachers and peer-reviews, and
  - based on what you learn from implementing the system.
- A presentation and demonstration (30 minutes/group) on March 5
  - the schedule will be decided later