Functional Analysis – functional allocation to components

Lecture 8

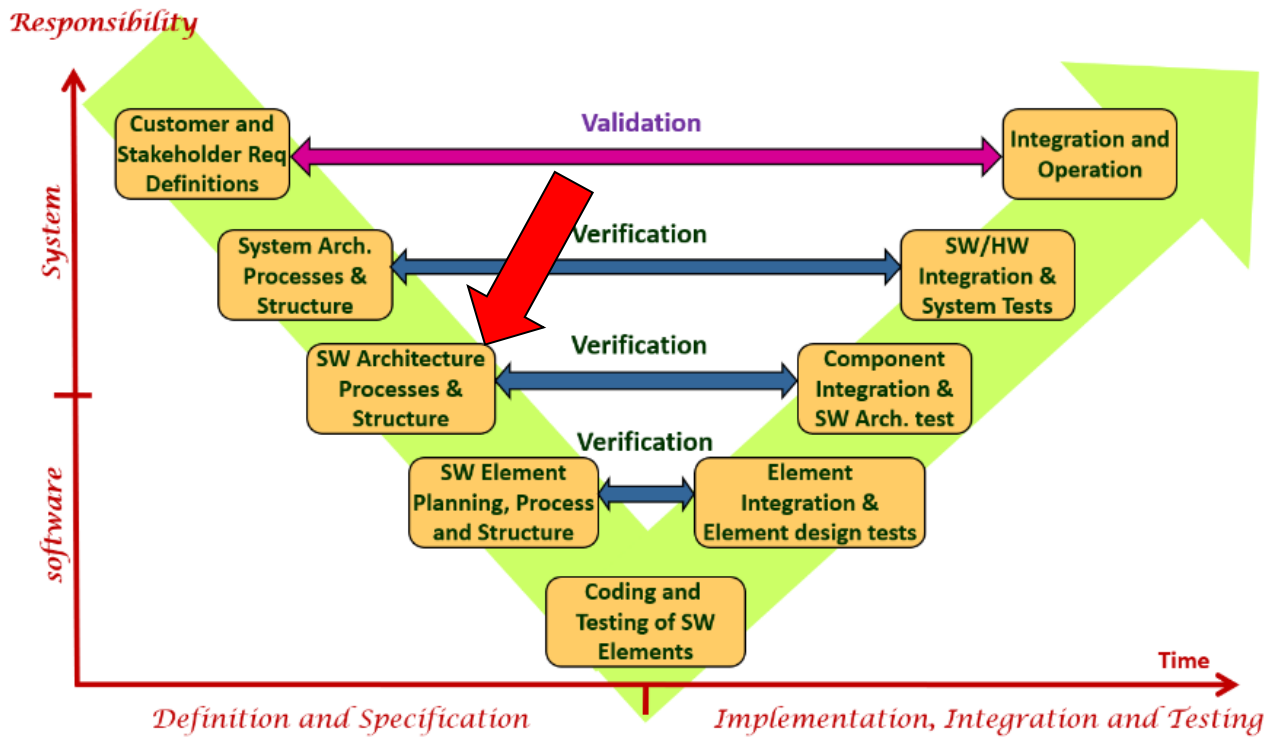
22 May 2025

Slides created by
Prof Amir Tomer
tomera@cs.technion.ac.il

Image source: ID 129218266 © Eveleen007 | Dreamstime.com

Functional Analysis

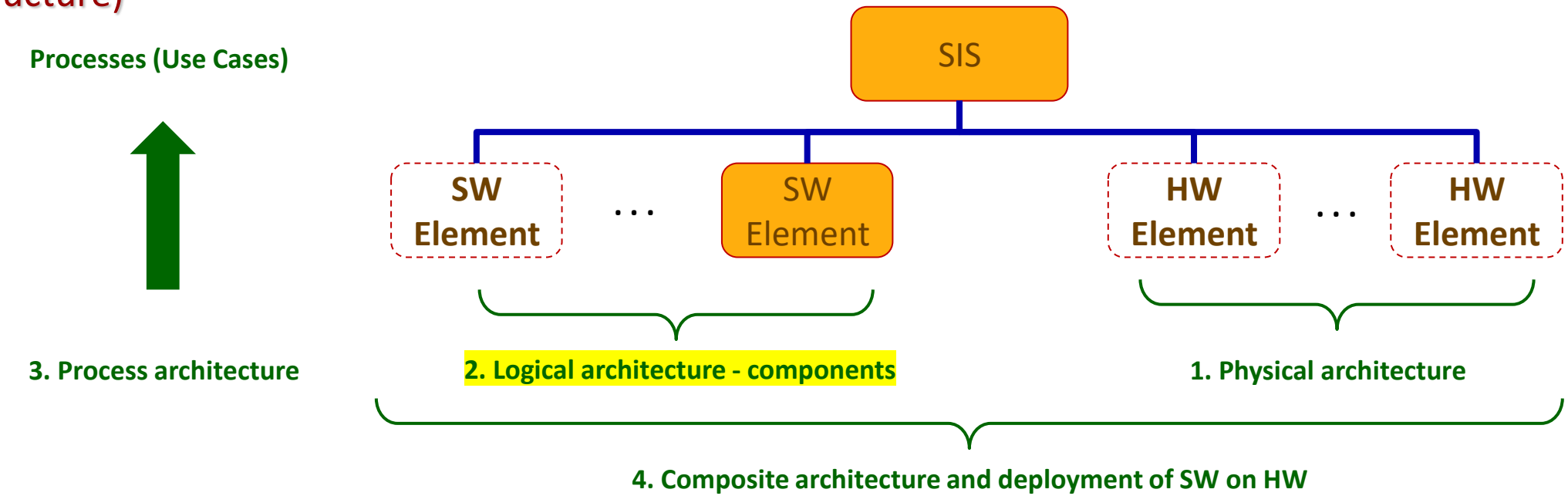
- Our goal: Define a set of functional elements (software components) to implement the system and assign them roles
- Inputs:
  - System processes (UC)
  - Functional requirements from the requirements table
  - Activity diagrams
- Outputs:
  - Software components list for the system



System Architecture: SIS architecture includes

Reminder

1. Physical architecture: Hardware components and physical connections – static model (structure)
2. Logical architecture: Software components and logical connections – static model (structure)
3. Process architecture: Implementation of processes via interaction between components - dynamic model (behavior)
4. Composite architecture: Implementation of logical connections via physical connections – static model (structure)



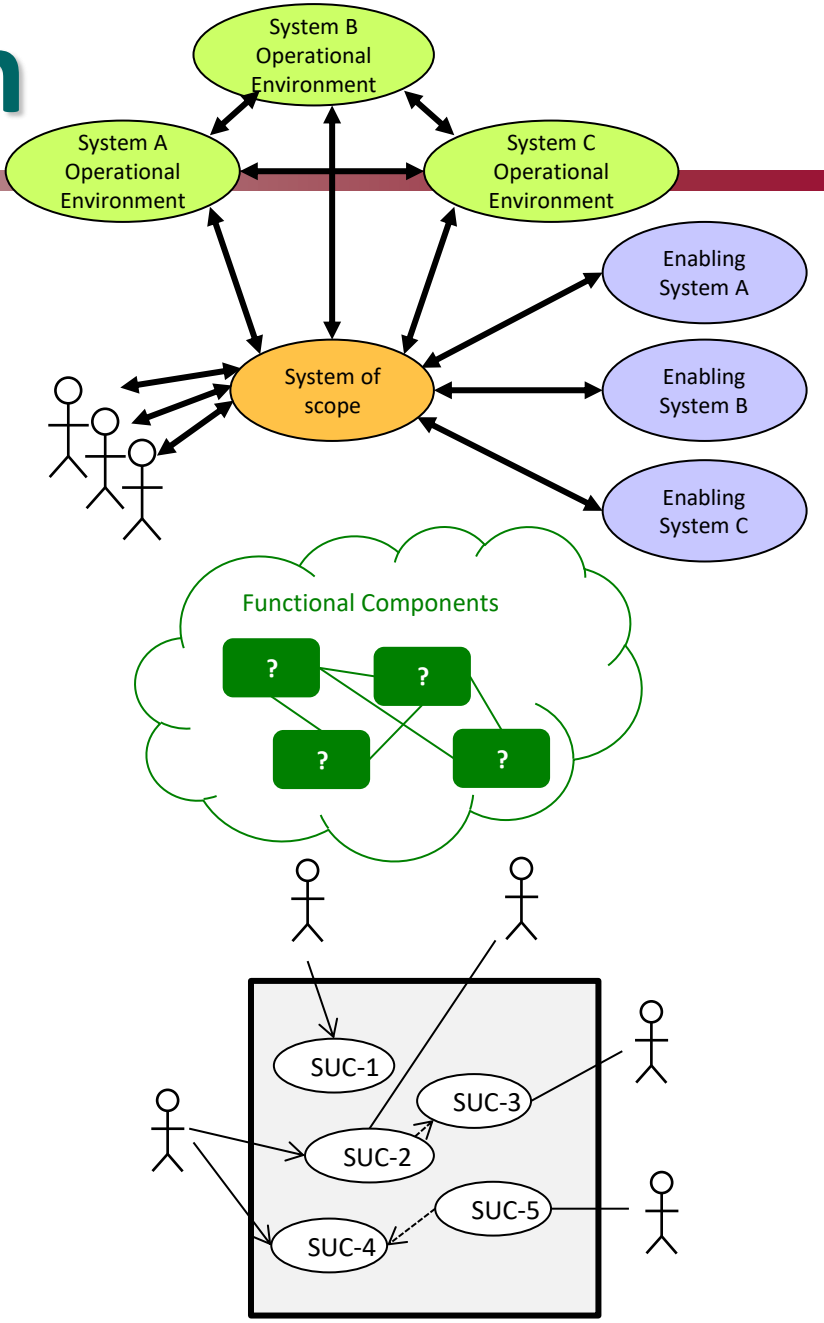
System Functional Breakdown

Until now we were at the system level

- Defined system processes through their interaction with the environment (use cases)
- All operations were done by “the system”

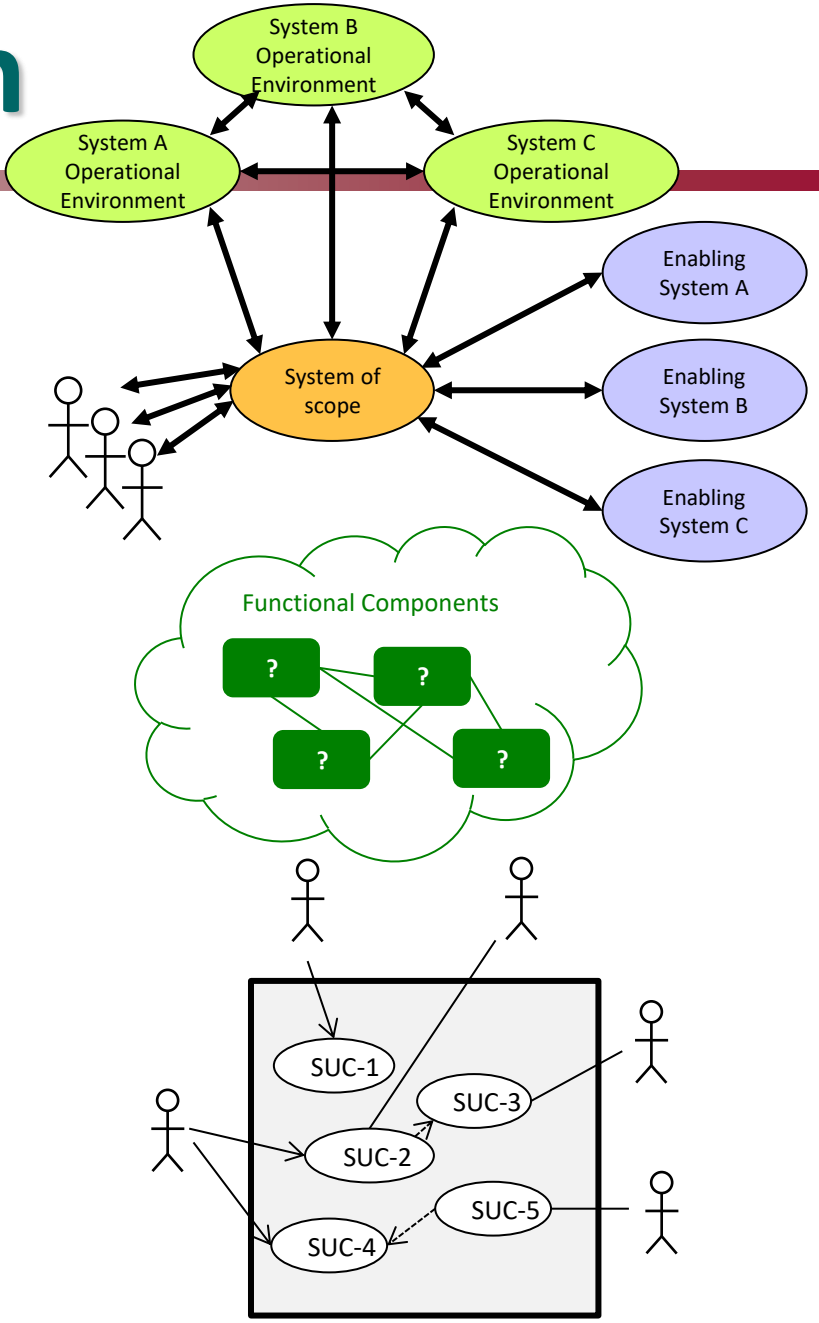
System functional breakdown

- System elements have specific functionality
- Elements perform activities via interactions with **other elements** or with **the environment**.



System Functional Breakdown

- How is functionality divided up among elements?
  - In hardware-based systems, a few options:
    - Hardware is specialized (e.g. wheels, engine, heating element)
    - Specialized hardware can do only specific tasks
  - SIS have more flexibility for functional assignment
    - Most elements are software
    - Software modules are installed on general purpose computers
    - Deployment of the software modules in a different manner can lead to different architectures.



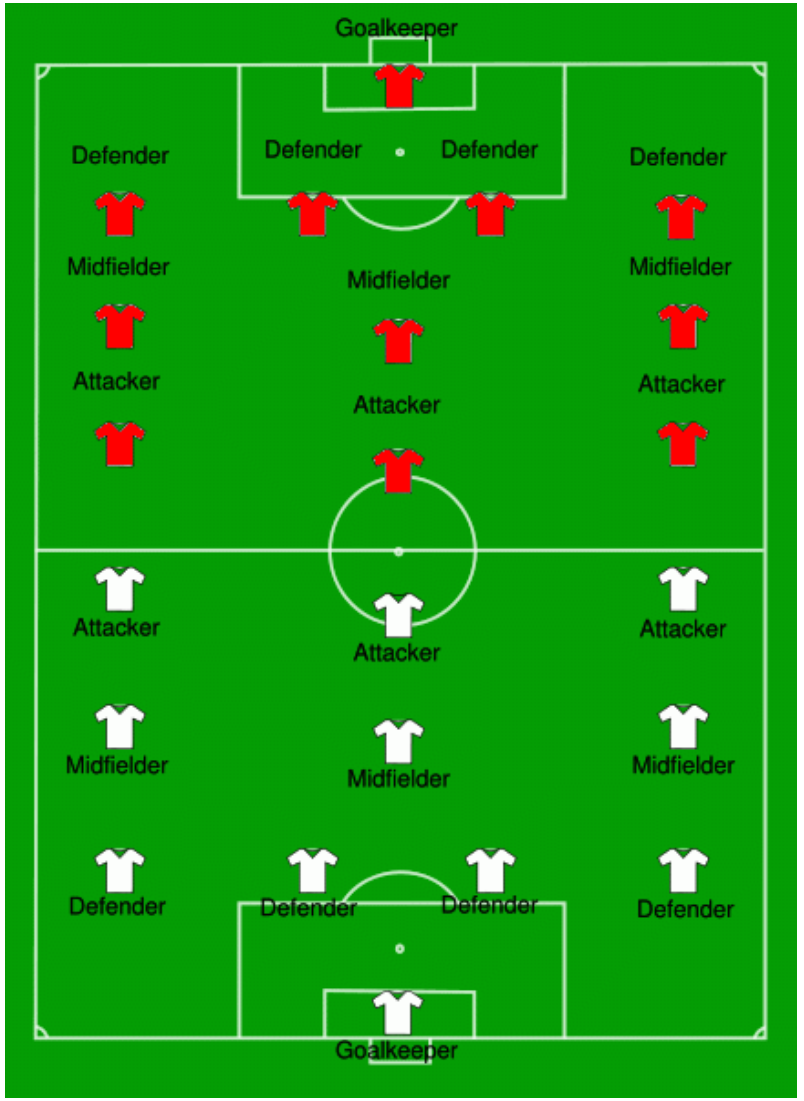
Functional Analysis

Functional analysis is process of:

- Identifying functions required to fulfill all tasks
- Choose elements (“recruit a task force”)
- Assign them the required functionality

For Soccer Team Use Cases

- **Attack**: Put the ball in the opposing team’s goal
- **Defense**: Prevent attacks from the opposing



22 May 2025

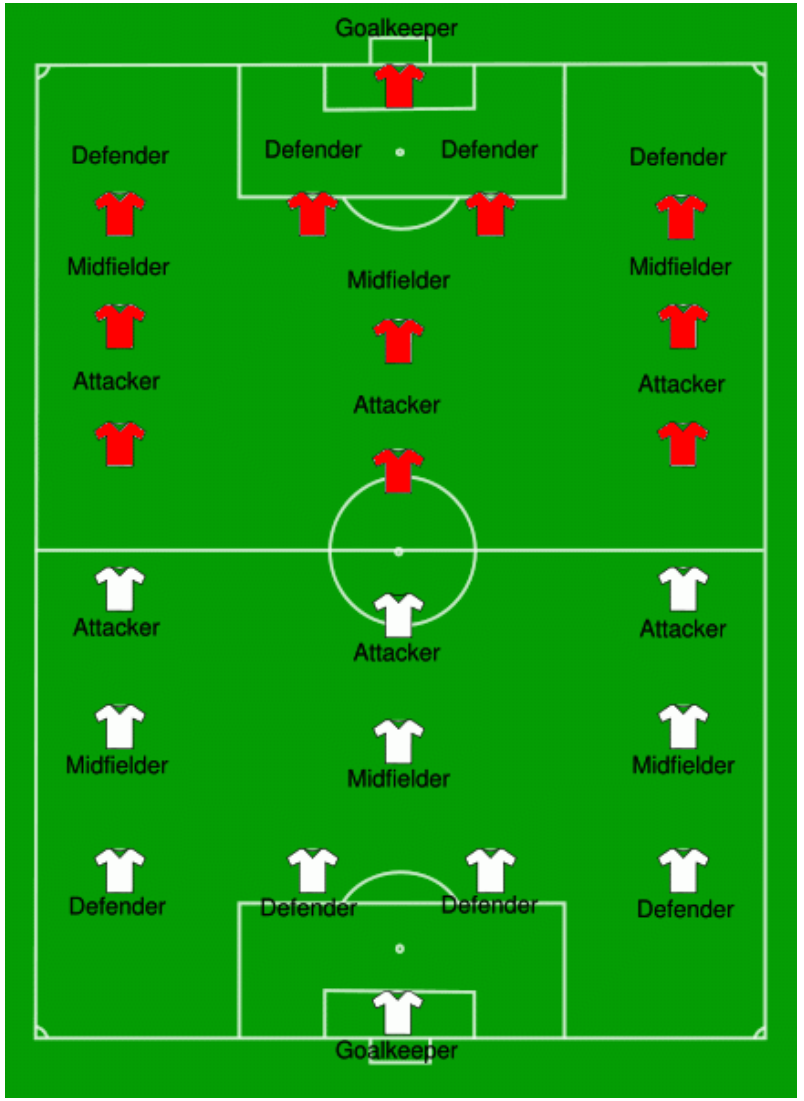
Functional Analysis

Required functionality:

- Kick to the goal
- Move the ball to the opposing team's side
- Overcome the opposing defenders
- Prevent the ball from entering the team's side
- Stop attacks
- Stop the ball

Functional elements (the team)

- Goalkeeper
- Defender
- Midfielder
- Attacker

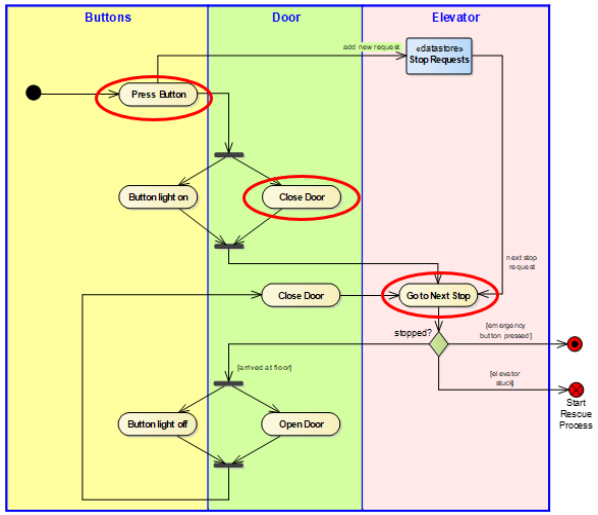


Where is the functionality?

- In things we created before:

Requirement	Type
…	
A passenger who is on a particular floor…presses the button for the direction he wants to travel	OR
The floor button turns on after being pressed if it was not already on	OR
After pressing the button, an elevator car traveling the appropriate direction will stop at the floor	OR
When the car arrives at the floor, the door opens, and the floor button turns off	OR
…	
A passenger inside the car who wants to ride to a particular floor presses the button for the floor	OR
The button in the car turns on after being pressed if it was not already on	OR
When the elevator stops at a floor, the door opens, and the floor button turns off	OR
If the car gets stuck during travel, the passenger can call for help	OR
The system is checked by a certified technician	OR
If the technician finds a problem, he attempts to fix it	OR

SUC-1	Call Elevator
Actors and Goals	Passenger: To receive an elevator car available for travel
Stakeholders and Interests	None
Pre-conditions	• User is on a floor in the building with an elevator door • System is operational (post-condition of UC: Start-up)
Post-conditions	An elevator car is at the user's floor with the door open (destination floor)
Trigger	**Passenger** pushes the up or down button on the floor
Main Success Sequence (MSS)	1. The system records the button press 2. The button lights up 3. The system finds a car traveling in the desired direction 4. The system assigns a stop for the car 5. The elevator arrives at the floor 6. The door opens 7. The floor button turns off



Requirements table

- Operational requirements

Use cases

- Operations performed during the use cases

Activity diagrams

- Action nodes

Finding functional requirements in the OR

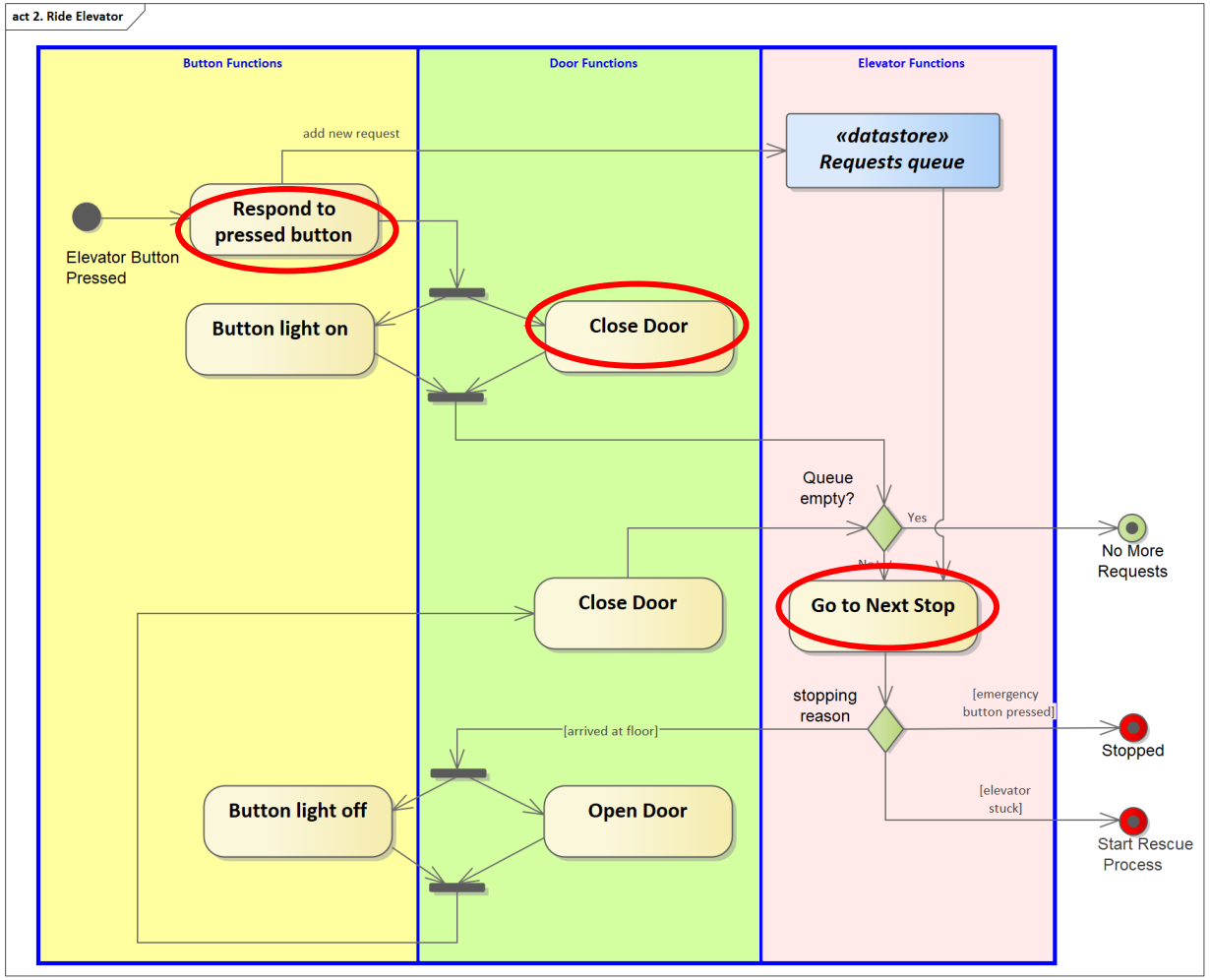
Requirement	Type
…	
A passenger who is on a particular floor…presses the button for the direction he wants to travel	OR
The floor button turns on after being pressed if it was not already on	OR
After pressing the button, an elevator car traveling the appropriate direction will stop at the floor	OR
When the car arrives at the floor, the door opens, and the floor button turns off	OR
…	
A passenger inside the car who wants to ride to a particular floor presses the button for the floor	OR
The button in the car turns on after being pressed if it was not already on	OR
When the elevator stops at a floor, the door opens, and the floor button turns off	OR
If the car gets stuck during travel, the passenger can call for help	OR
The system is checked by a certified technician	OR
If the technician finds a problem, he attempts to fix it	OR

Finding functionality in Use Cases

SUC-1	Call Elevator
Actors and Goals	Passenger: To receive an elevator car available for travel
Stakeholders and Interests	None
Pre-conditions	• User is on a floor in the building with an elevator door • System is operational (post-condition of UC: Start-up)
Post-conditions	An elevator car is at the user's floor with the door open (destination floor)
Trigger	**Passenger** pushes the up or down button on the floor
Main Success Sequence (MSS)	1. The system records the button press 2. The button lights up 3. The system finds a car traveling in the desired direction 4. The system assigns a stop for the car 5. The elevator arrives at the floor 6. The door opens 7. The floor button turns off

Finding functionality in Activity Diagrams

- Swim lanes (if defined) can be the basis for functional assignments (see later)



Functional Grouping

- Assign groups of similar functionality into similar modules

By the physical node where the functionality will take place (if known)

By the type of data processed

- Raw data
- Audio, video
- Photos

By the level of data processing performed

- Raw data collection
- Data import and categorization
- Analysis and decision making

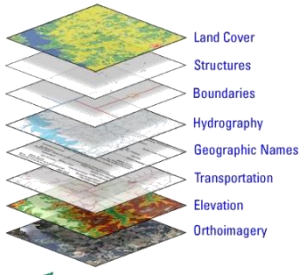
By the type of operations performed

- View/Display
- Data management
- Hardware control

Functional Grouping

- Dividing up by tiers by type of operations

Data Tier



- Operations connected to storage/retrieval of data
- Example: map-based retrieval

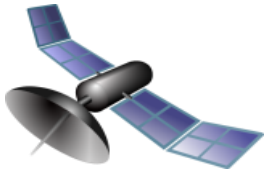
Application Tier

- Operations connected to system processing
- Example: tracking route

Presentation Tier

- Operations connected to input/output and user interaction
- Example: show location

Control Tier



- Operations connected to controlling external devices/drivers
- Example: GPS geolocation

In Class Activity: Functional Analysis and Assignment for the elevator

- I assume you know the Activity Diagrams, Use Cases, and Requirements for the elevator story
- What to do:
  - Create a list of system functions
  - Divide the list into two groups
    - Operating a single elevator
    - Operating the complete system

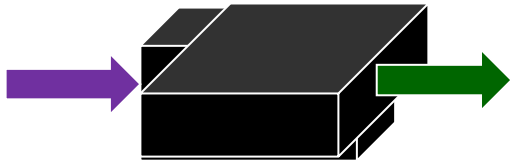
Functional components

Functional components (logical, software)

- Performs 1+ operations/services
- Has 1+ interfaces through which other components or external entities can start it
  - With/without sending information
- Run operations on other entities or components
- Doesn't constrain implementation choices (many ways to implement)

Component is a black box to others

- Outsiders use I/O to probe component's functional readiness
- Can measure its performance externally only
- Can be replaced with other components with identical I/O specs and behavior



22 May 2025

Efficient assignment of functionality to components

- A component will perform its tasks better the more **specialized** and **independent** it is
- Specialization and independence are described with

Tight Cohesion

- All functionality in the component have something in common
  - Per properties of group assigned to it
- Don't assign unrelated functions
  - Don't try to "keep it busy"

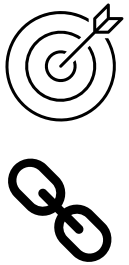
Loose Coupling

- Minimal dependency between components
- Not affected by changes in other components
- In SW: encapsulation and information hiding

Image source: Wannapik (https://www.wannapik.com/vectors/81795#!)

Fine Tuning Functional Assignment and Assignment to Components

- Based on the functional assignment before, assign system functions to 2 components
  - Component to run a single elevator car (5 functions)
  - Component to run the whole system (10 functions)
- Based on tight cohesion and loose coupling, we can tune the assignment a bit
  - 10 is a lot of functionality for one component
  - For each sub-group of functionality, define a component
  - **→Functional assignment**

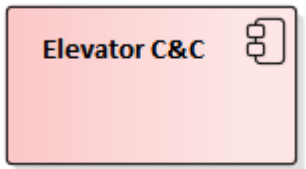


22 May 2025

Refining the elevator example

Single car operations

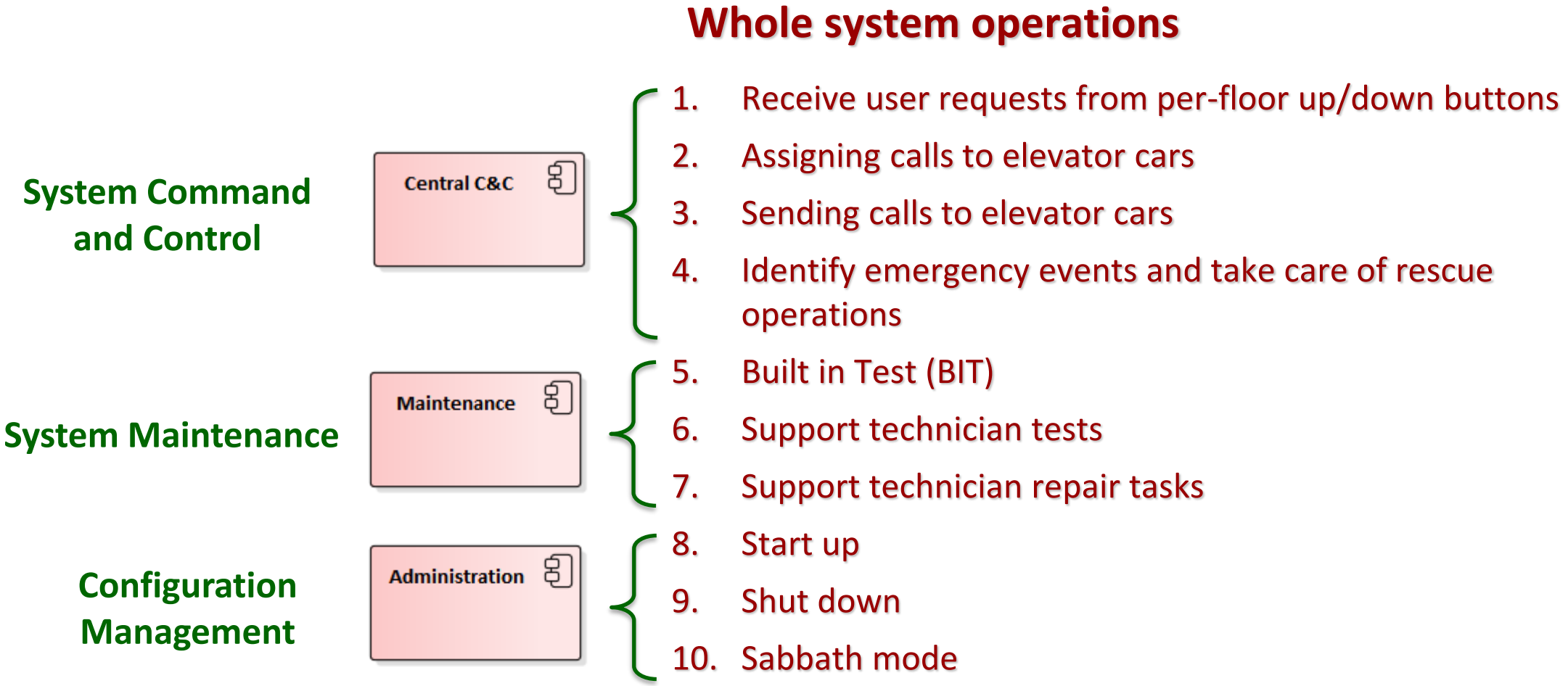
1. Receive user requests from in-car floor buttons
2. Receive requests from central operations
3. Managing travel plan
4. Engine control (travel, stop)
5. Door control (open, close)



Command and Control for Single Car

22 May 2025

Refining the elevator example



In class assignment: Functional Assignment and Components

1. List the functionality in the use cases from **ePark**
2. Group the functionality to 2-4 functional groups
3. Refine the assignment into 6-8 functional components
4. Create a new component diagram and put the components in it
   - Divide the components into groups using swim lanes

Conclusion

- Functional Analysis