Use Case Identification and Specification

Lecture 4
10 April 2025

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

4.4.1 User searches for a farm

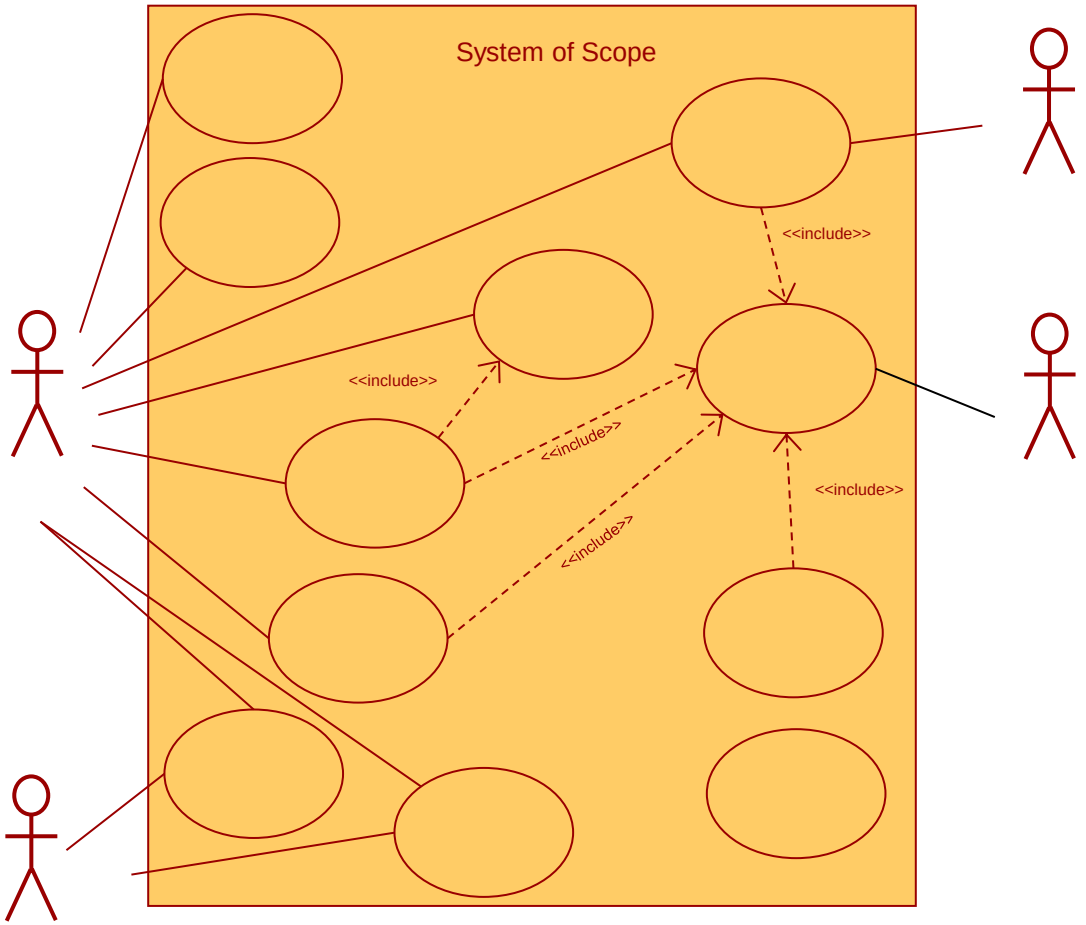
Actors	• Support user: Primary Actor – wants to locate a specific farm.
Other Stakeholders	
Pre-conditions	• User opened the web app in his browser and web app is online.
Post-conditions	• User views farm details screen of the farm he searched for.
Trigger	User clicks on the search bar
MSS	1. User starts typing either farm name or farm code in the search bar 2. UI shows auto-complete options in a drop list while user keeps typing 3. User sees the farm that he wants to view and clicks on its name in the drop list or presses 'Enter' on keyboard
Branch A	Exception in step 3: 1. User inputted incomplete farm name/code or name/code that does not exist and pressed 'Enter' 2. Search bar displays for the user "No farms found"
Requirements Trackback	Functional: 4

Topics for today

- Use cases
  - Identifying system use cases
  - Use Case Definition and diagram
  - Specifying use case contents

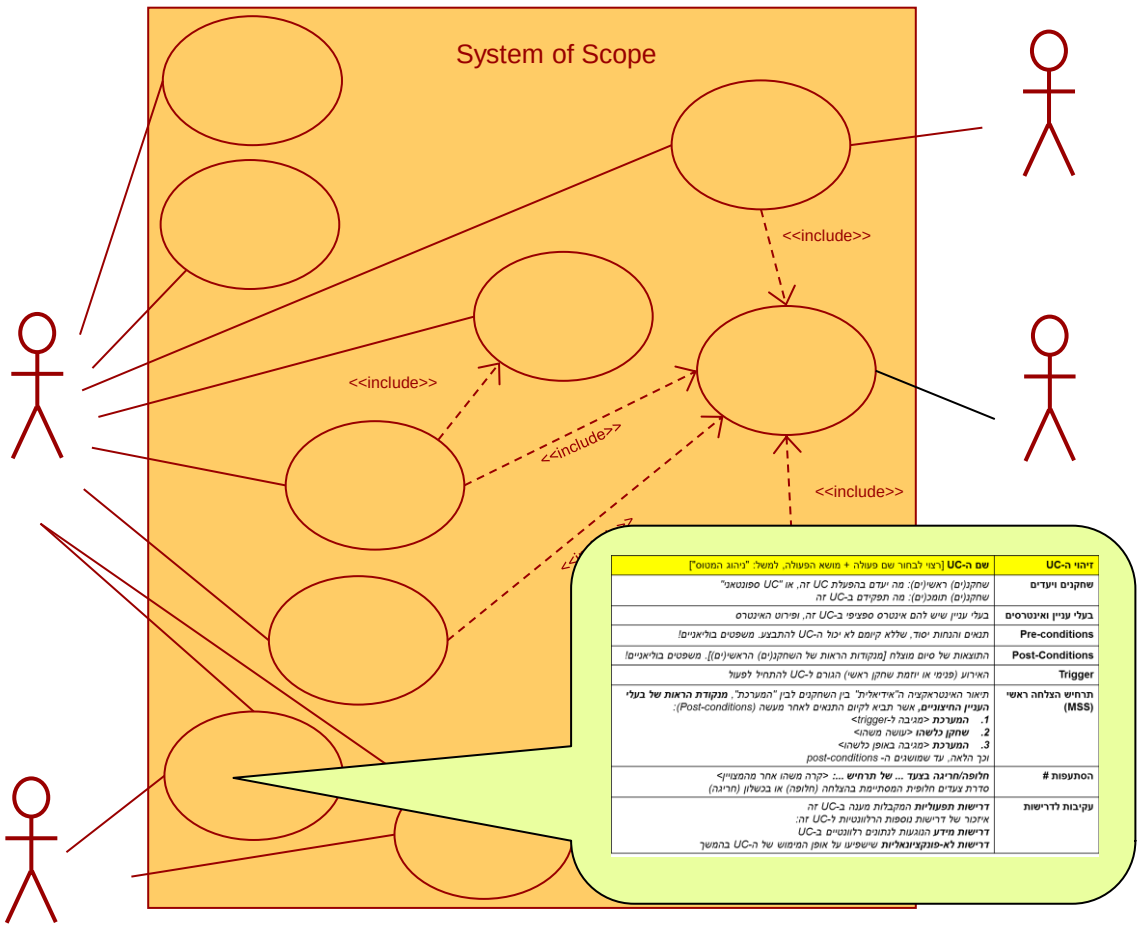
Use Case Model: Describing System Use Cases

- Use Case Diagram: Shows the whole system and its environment
  - Rectangle the depicts the system boundary
  - Stick figures: actors – any entity that is not part of the system but interacts with it
  - Ellipses: Use cases – the services and processes of the system
  - Connections between actors and use cases
  - Connections between use cases



Use Case Model: Describing System Use Cases

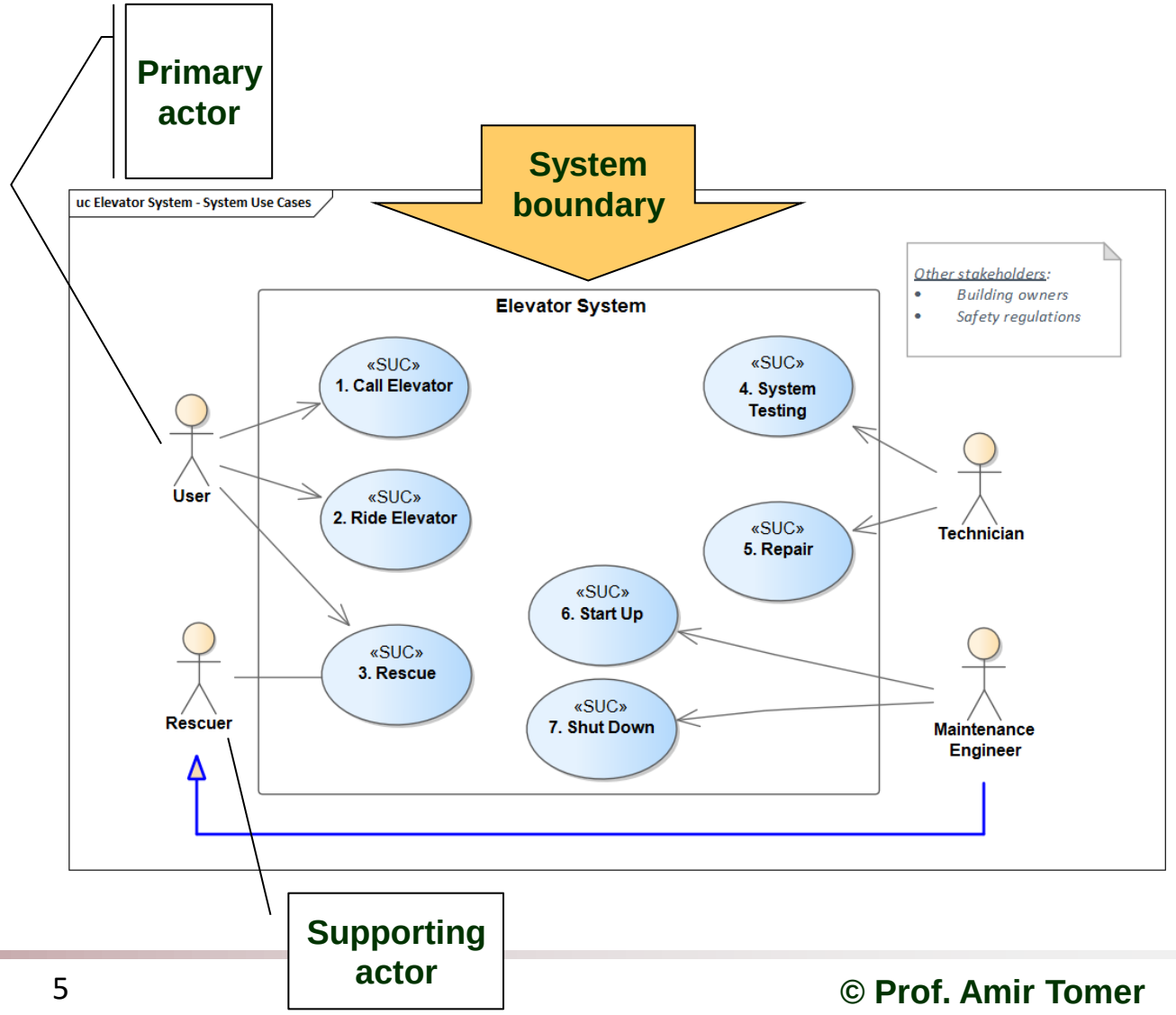
- Use Case Specification:
  Structured text for each UC
  - Stakeholders and actors for the UC
  - Conditions for the UC to occur
  - Expected outcome for the UC
  - Ways for the process to occur (may lead to different results)



זיהוי ה-UC	שם ה-UC [רצוי לבחור שם פעולה + מושא הפעולה, למשל: "ניהול המצוס"]
שחקנים ויעדים	שחקן(ים) ראשי(ים): מה יעדם בהפעלת UC זה, או "UC ספונטאני" שחקן(ים) תומכ(ים): מה תפקידם ב-UC זה
בעלי עניין ואינטרסים	בעלי עניין שיש להם אינטרס ספציפי ב-UC זה, ופירוט האינטרס
Pre-conditions	תנאים והנחות יסוד, שללא קיומם לא יכול ה-UC להתבצע. משפטים בוליאניים!
Post-Conditions	התוצאות של סיום מוצלח [מנקודות הראות של השחקן(ים) הראשי(ים)]. משפטים בוליאניים!
Trigger	האירוע (פנימי או יחמת שחקן ראשי) הגורם ל-UC להתחיל לפעול
תרחיש הצלחה ראשי (MSS)	תיאור האינטראקציה ה"אידיאלית" בין השחקנים לבין "המערכת", **מנקודת הראות של בעלי העניין החיצוניים**, אשר תביא לקיום התנאים לאחר מעשה (Post-conditions): 1. **המערכת** <מגיבה ל-trigger> 2. **שחקן כלשהו** <עושה משהו> 3. **המערכת** <מגיבה באופן כלשהו> וכך הלאה, עד שמושגים ה- post-conditions
הסתעפות #	**חלופה/חריגה בצעד ... של תרחיש ...:** <קרה משהו אחר מהמצויין> סדרת צעדים חלופית המסתיימת בהצלחה (חלופה) או בכשלון (חריגה)
עקיבות לדרישות	**דרישות תפעוליות** המקבלות מענה ב-UC זה איזור של דרישות נוספות הרלוונטיות ל-UC זה; **דרישות מידע** הנוגעות לנתונים רלוונטיים ב-UC **דרישות לא-פונקציונאליות** ששיפיעו על אופן המימוש של ה-UC בהמשך

UML Use Case Diagram: Visually depicting system processes

- System Use Cases (SUC) for system services
- System environment: actors and other stakeholders
  - Primary actors: Initiate services (e.g. user)
  - Supporting actors: Help the system offer its services (e.g. rescuer)
  - Stakeholders: Have an interest, but don't participate (e.g. safety policies)



Documenting the Figure: Stakeholders

Name	Role
User	Uses the elevator to travel to floors in the building
Rescuer	Helps rescue passengers from stuck elevators
Technician	Checks the system and fixes as needed
Maintenance Engineer	Starts and stops the system
Building owner	Gains benefit from the system and defines limitations on the systems operation
Safety regulations	Required safety rules for operating and using the system

- The list of actors and stakeholders (system environment)

Documenting the Figure: Use Cases

ID	UC Name	Main actor(s)	Description
SUC-1	Call Elevator	User	Calls the elevator to a floor
SUC-2	Ride Elevator	User	Travel in an elevator from floor to floor
SUC-3	Rescue	Rescuer	Rescue from a stuck elevator
SUC-4	System Testing	Technician	Checking the system
SUC-5	Repair	Technician	Fixing the system
SUC-6	Start Up	Maintenance Engineer	Starts up the system
SUC-7	Shut Down	Maintenance Engineer	Shuts down the system

- The list of processes the system can perform

Sample List of Use Cases in ePark System

Initiator	Use Case	Goal
Guardian	Register and Check-in	הכנסת הילד לפארק, מסירת פרטים, פתיחת כרטיס אלקטרוני וחשבון מלווה וקבלת צמיד
Guardian	Manage eTicket	רכישה וביטול של כניסות למתקנים
Guardian	Park Check-out	סגירת חשבון והחזרת הצמיד
Guardian	Child Tracking	מעקב שוטף אחר מיקומו של הילד בפארק
Spontaneous	eTicket Update	עדכון הכרטיס האלקטרוני כחלק מתהליכים אחרים (included UC)
Child	Enter Attraction	כניסה למתקן שעשועים בכפוף להרשאות ולמגבלות
Child	Exit Attraction	יציאה ממתקן שעשועים לאחר סיום השימוש או כתוצאה מתקלה
Supervisor	Attraction Setup	קביעת ההגדרות והמגבלות של מתקן שעשועים
Supervisor	Attraction Monitoring	מעקב שוטף אחר פעולת המתקנים וביצוע פעולות הפעלה/הדממה בהתאם לצורך
Spontaneous	Attraction Breakdown	השבתת מתקן וביצוע הפעולות הנדרשות כתוצאה מתקלה
Supervisor	Start Up	בדיקת תקינות הפארק ואיתחול פעילותו
Supervisor	Shut Down	כיבוי כל מתקני הפארק והפסקת פעילותו

Exercise: Build a Use Case Diagram

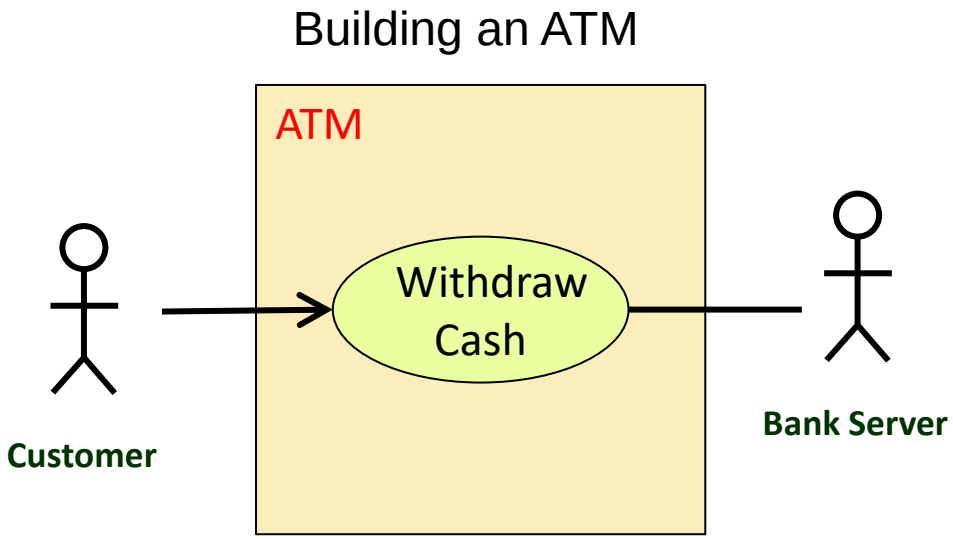
- Build a use case diagram for the ePark system
- Include all use cases identified in the previous exercise
  - Draw the system boundaries
  - Put the main actors (external entities that initiate actions) outside the system boundary
  - Put the processes (use cases) identified in the boundary
  - Connect arrows between the main actors and each use case they initiate
  - If there are supporting actors (external entities that help a process work)
    - Put them outside the boundary
    - Connect them with an association line (not an arrow) to the use case

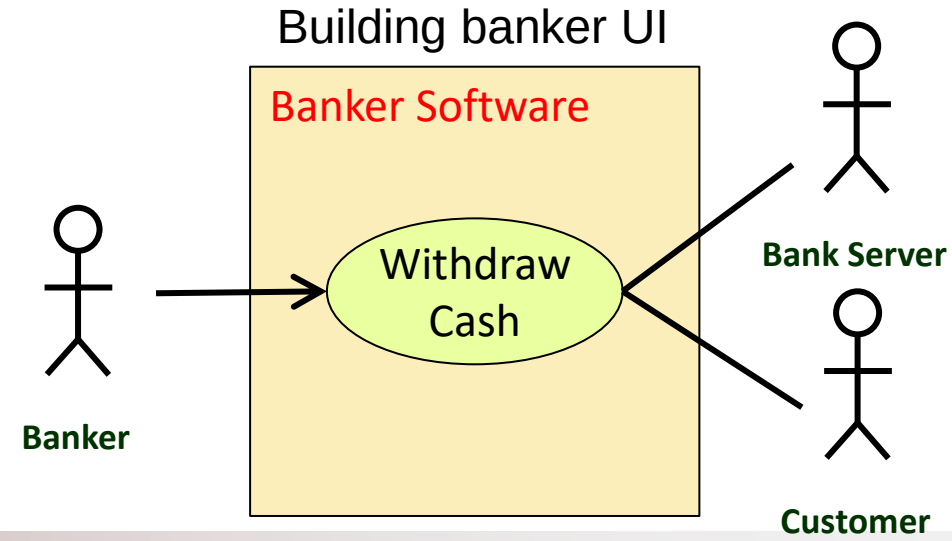
So Far

- Use cases
  - Identifying system use cases
  - Use Case Definition and diagram
  - Specifying use case contents

System Boundaries (or, What is the system?)

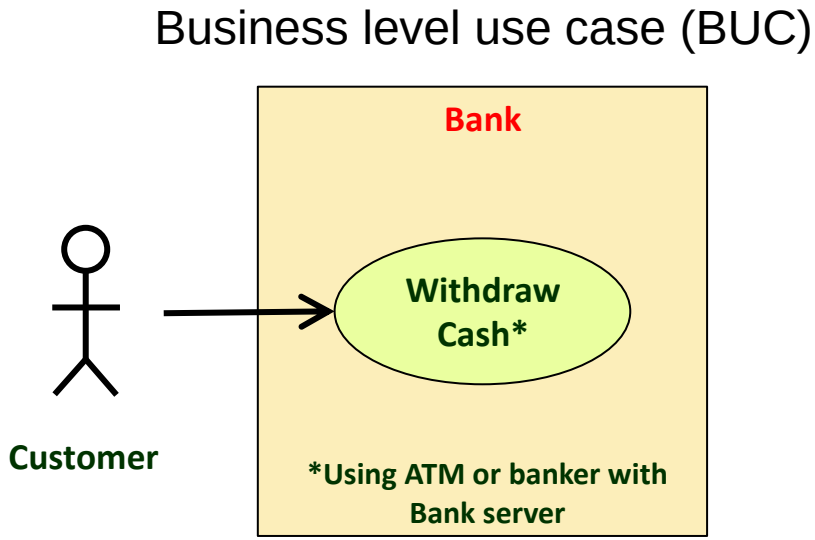
- The system boundaries define the difference between the system under development and its external environment
  - Use cases may occur differently depending on how the system is defined
  - Use cases are within the system, actors are outside





System Boundaries (or, What is the system?)

- The system boundaries define the difference between the system under development and its external environment
  - Use cases may occur differently depending on how the system is defined
  - Use cases are within the system, actors are outside



Definition: Use Case

- A complete task performed by the system with the goal of reaching a defined and measurable outcome for one or more primary actors or stakeholders

A use case can start when an actor initiates it or when the system initiates it

- Spontaneous use case: A use case initiated by the system itself

Use case is an agreed upon set of interactions between the system and its actors

- Interaction and its outcomes are visible to the actors and stakeholders

Use cases include all potential outcomes from the interaction while trying to reach the goals, even if goals are not achieved

Image sources: https://sc02.alicdn.com/kf/H9e19c29130ae42a9a5d76644294855 8e8.jpg_350x350.jpg, Image by Pete Linforth from Pixabay

Use Case Contents

- Use case specifications: What's in a use case

Actors	Stakeholders	Goals and interests	Requirements trackback
• Primary • Supporting			

- Definition of how events can unfold

Potential paths	Potential outcomes
	• Success – the use case achieved its goals • Failure – the use case didn't achieve all goals (maybe it got some)

Use cases and actors

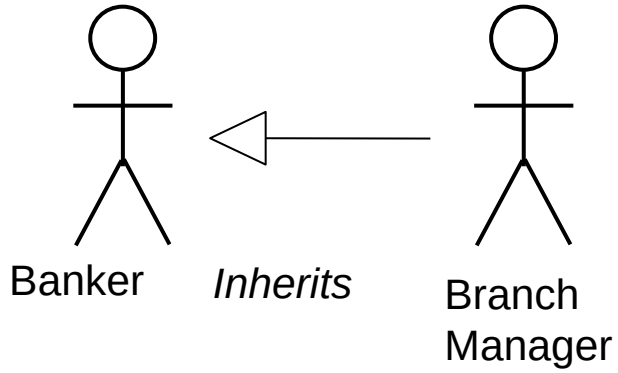
- Every use case has 0 or more actors

Primary Actor	Initiates the use case
	Goal: to achieve some results
	Initiation happens through a system PROVIDED interface
Supporting actor	Doesn't start, but the system communicates with or uses the actor
	Goal: to help the system provide its service
	Interaction happens via some system REQUIRED interface
Spontaneous use case	No primary actor
	The system starts the use case due to an internal event or decision

More on Use Cases and Actors

Inheritance

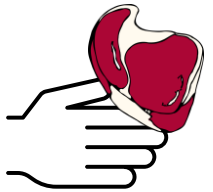
- An actor can “inherit” from another actor (like classes)
  - A inherits from B = A can do everything B can PLUS other stuff



Stakeholders and Use cases

- Aside from actors, use cases can be assigned to stakeholders (or stakeholder groups)
- Stakeholders have a specific interest in what happens in the use case

About Stakeholders



- **Stakeholder**: A person, team, organization, or other entity that is not an actor but has interests or concerns about how the system works
- Interests include:
  - Receiving services from the system
  - Giving services to the system
  - Performing operations for its own needs (sometimes to help others)
  - Monitoring the quality of the system's operations
  - Being influenced by the system's operations
  - Ensuring the system respects rules, standards, and regulations
- Stakeholders' interests are fulfilled during the use case
  - E.g. charging a fee for banking transactions (for the bank)
- Common stakeholders
  - Users (who are not active in the current UC)
  - Technicians, Support staff
  - Administrators
  - Regulatory bodies or experts who have an interest in the system (e.g. safety, ergonomics, data security)
  - Management (want metadata)

Image source: https://pixy.org/4589807/

Is a burglar an actor in a home security system?

A burglar sets off the home security system, so is he an actor?

Many things might set off the alarm (cat, curtain, lightning, dirt)

Are they all actors?

The burglar doesn't set off the alarm on purpose to achieve a goal

- So the alarm system sets itself off
- An internal operation

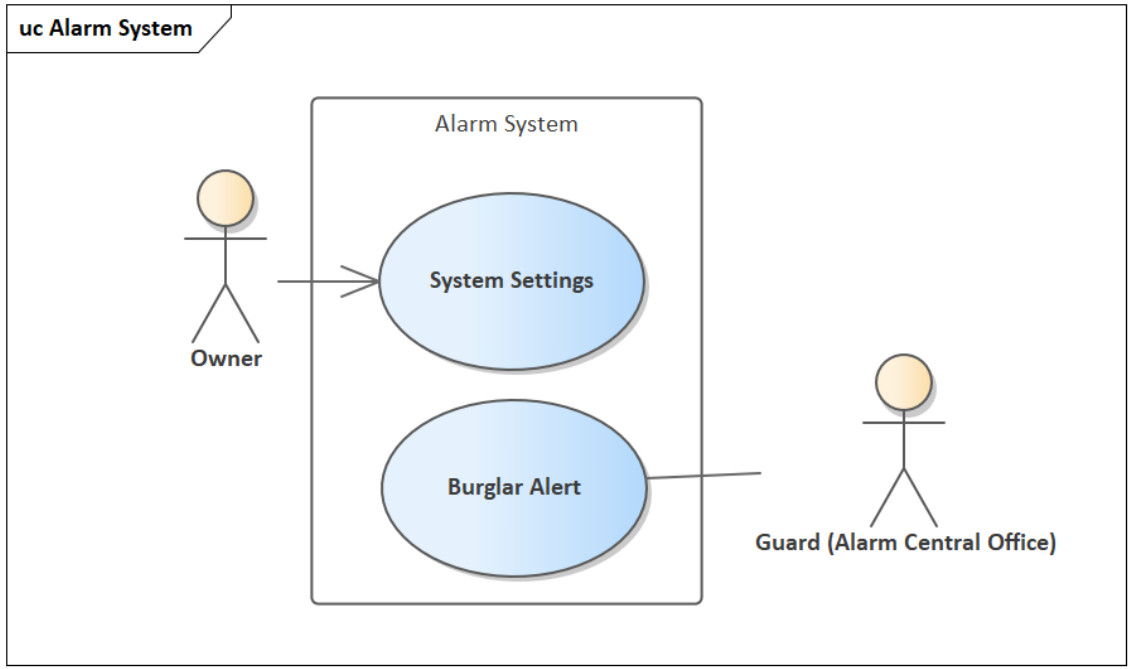
The alarm service isn't there to serve the burglar –it's there for the homeowner.

- The homeowner is a stakeholder is the "alarm" use case – to achieve her goals

Homeowner might be an actor in other use cases of the system – e.g. alarm configuration

Use case diagram for an alarm system

- Burglar alert is a spontaneous use case
- Burglar doesn't appear in the use case diagram
  - The situation can arise even without a burglar there

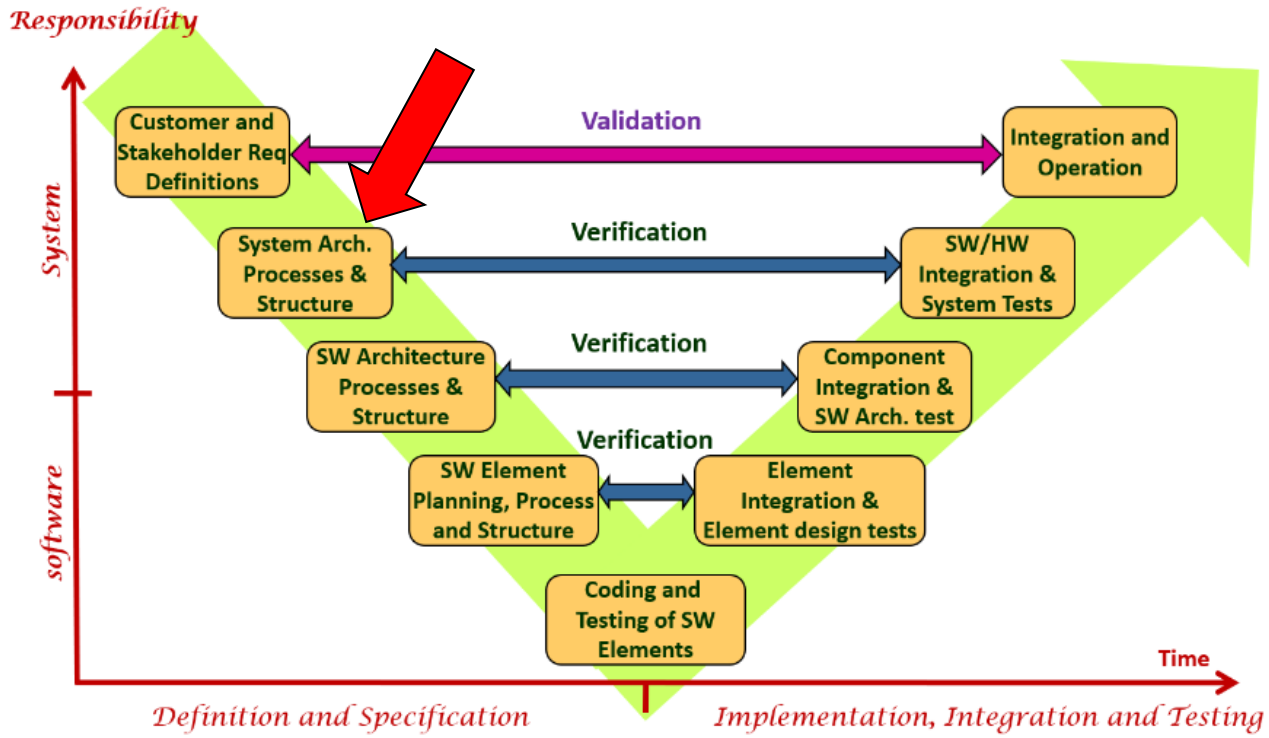


So Far

- Use cases
  - Identifying system use cases
  - Use Case Definition and diagram
  - Specifying use case contents

Specifying Use Case Text

- Our goal: Specify the conditions for and method by which the system operates
- Inputs:
  - Technical specification
  - Operational specification
  - Requirements table
  - Use case list
- Outputs:
  - Use Case details
  - Trackback to requirements



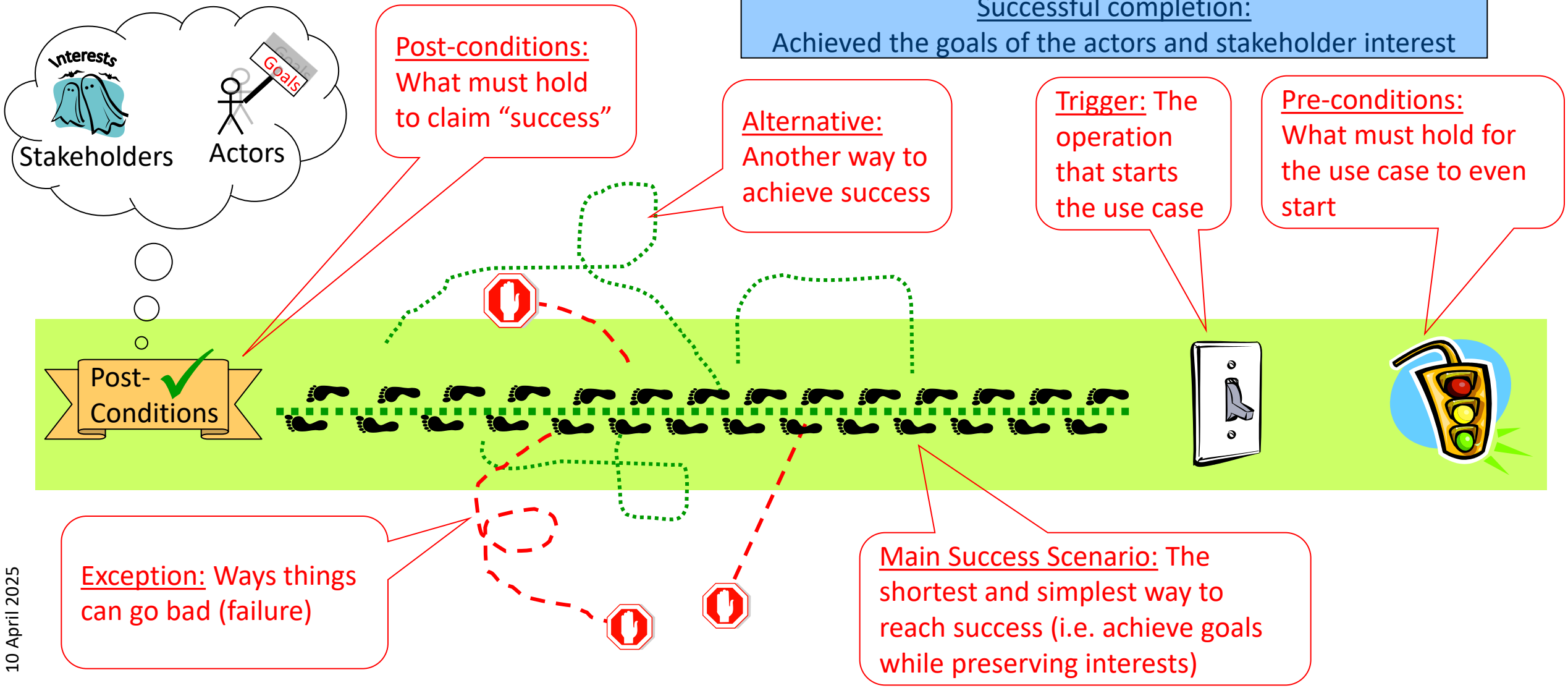
Discussion

- Given a system process in a navigation app (e.g. Waze, Google Maps):

Initiator	Use case	Goal
Driver	Reporting adverse conditions	Reporting an adverse conditions on the road and sending it to the server

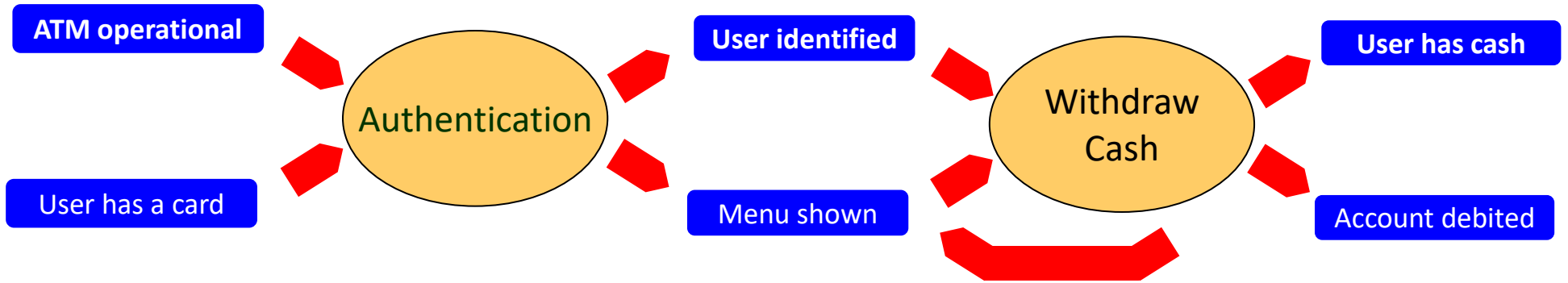
- Questions:
  - Are there any situations when the use case can't be performed? If yes, when?
  - How does the user start the use case?
  - How does the use case proceed? What interaction steps take place?
  - What defines whether the use case succeeded or not?
  - Can the use case complete in different ways?
  - Are there other stakeholders who aren't part of the use case who will benefit from it?

Parts and Procedure of the Use Case Specification

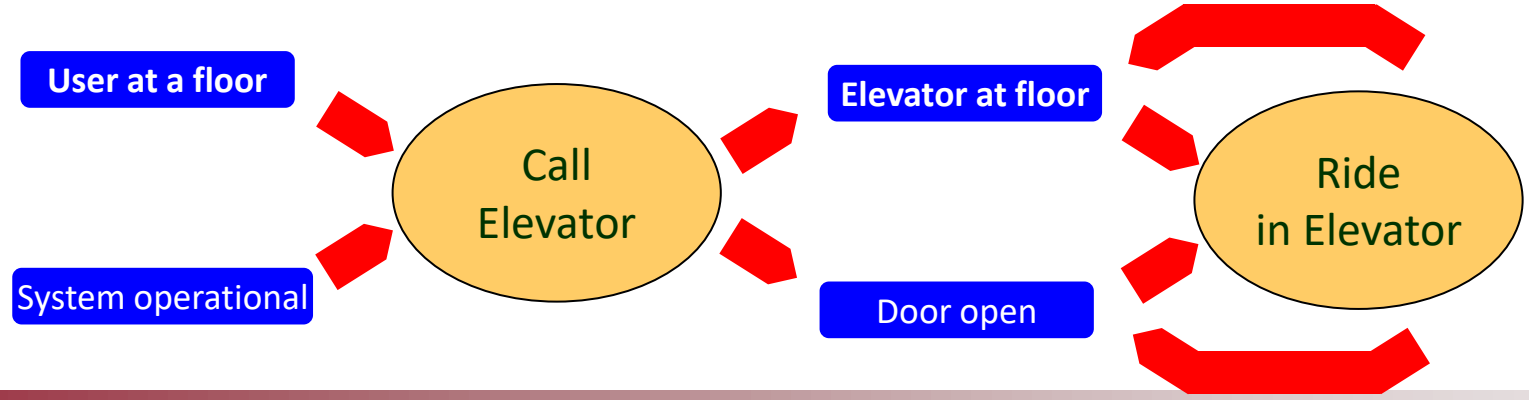


Connections between Use Cases

- Post-conditions of one use case are the pre-conditions of another



- Flow of operations is not fixed (events occur as conditions enable)
- Preconditions can be fulfilled due to another use case occurring



Format for a textual use case specification

Use Case ID	Use Case name (choose an operation and subject, e.g. “flying the plane”)
Actors and Goals	Primary actor(s): What are their goals in performing the use case, or “Spontaneous” Supporting actor(s): What are their roles in the use case
Stakeholders and Interests	The stakeholders who have a specific interest in the use case, list their interests
Pre-conditions	Conditions and basic assumptions that must hold to perform the UC. Ensure the conditions can be only true or false!
Post-conditions	Results of a successful UC run from the perspective of the primary actor(s). Write unambiguous, true or false statements.
Trigger	The event (primary actor’s actions or internal event) that lead to the use case starting
Main Success Sequence (MSS)	The “ideal” interaction between the actors and the system from the perspective of the stakeholders. The series must lead to the fulfillment of the post-conditions. 1. The system <responds to the trigger> 2. An actor <does something> 3. The system <responds somehow> And so on until the post-conditions are reached
Branch #	Alternative/Exception from step … of … <something different happened> Set of steps that leads to the use case succeeding or failing, perhaps returning to the MSS
Requirements trackback	Operational requirements: that correspond to the use case Other related requirements: Data requirements that connect to data in the use case Non-functional requirements that affect the use case implementation

Use cases can have loops and conditionals (if-then), but not branches (if-then-else)

Branches can be generic at first and have the details filled in later

SUC-1 Call Elevator

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
Branch A	Alternative from step 2 of MSS: The button is already lit 2A1. Return to step 5 of the MSS.
Requirements trackback	…

These steps occur in parallel to the rest of the UC,
We'll show how to model this later.

SUC-2 Ride Elevator

SUC-2	Ride Elevator
Actors and Goals	Passenger: To arrive at the desired floor
Stakeholders and Interests	Safety standards: Ensure passengers are not stuck in elevators
Pre-conditions	• User is in an elevator car • System is operational (post-condition of UC: Start-up)
Post-conditions	• Car arrives at the desired floor (destination) • Passengers can leave the elevator (interest)
Trigger	**Passenger** pushes the button for the desired floor
Main Success Sequence (MSS)	1. The car records the button pressed and adds a scheduled stop at the floor 2. The button lights up 3. The car door closes (if it was open) 4. The car continues traveling to the next floor in its stop list 5. The car stops at the floor 6. The door opens 7. The floor's light turns off 8. Return to step 3.

These steps occur in parallel to the rest of the UC,
We'll show how to model this later.

10 April 2025

SUC-2 Ride Elevator

SUC-2	Ride Elevator
Branch A	Exception from step 4 of the MSS: Passenger presses the emergency stop button 4A1. The car immediately stops 4A2. The car cancels all upcoming planned stops 4A3. End of use case
Branch B	Exception from step 4 of the MSS: Car gets stuck 4B1. Call for rescue is issued (transfer to SUC-3 that extends)
Branch C	Alternative from Step 4 of the MSS: There are no more floors in the stop list 4C1. The elevator remains at the floor. 4C2. End of use case
Requirements trackback	…

SUC-3 Rescue

SUC-3	Rescue
Actors and Goals	Passenger: To be rescued from a stuck elevator Rescuer: Supporting actor (helps rescue)
Stakeholders and Interests	Safety standards: Ensure passengers are not stuck in elevators
Pre-conditions	• Car is stuck (got stuck while traveling, extending SUC-2)
Post-conditions	• Passenger can exit the car (goal + interest)
Trigger	**Passenger** calls for rescue
Main Success Sequence (MSS)	1. The system calls the rescuer 2. The rescuer goes to the machine room and starts “rescue” mode 3. The elevator arrives at the ground floor 4. The door opens
Branch	None
Requirements Trackback	…

SUC-4 System Testing

SUC-4	System Testing
Actors and Goals	Technician: Perform a complete system test and ensure the system is in proper working order
Stakeholders and Interests	Passengers and Safety standards: System is safe and fit for use (reliability, availability, safety)
Pre-conditions	• System is operational (via SUC Start up) • System is not in use
Post-conditions	• All system functions are nominal • A complete functional report is recorded
Trigger	**Technician** starts the test program
Main Success Sequence (MSS)	1. The technician opens a new functional report 2. The technician orders a “going up” car to a floor that has not been tested yet, the car arrives (SUC-1) 3. The technician orders a “going down” car to the same floor, the car arrives (SUC-1) 4. The technician marks that the floor is nominal 5. The technician repeats steps 2-5 for each floor 6. The technician goes into a car that has not been tested yet 7. The technician uses the car to travel to every floor (SUC-2) 8. The technician marks that the car is nominal 9. The technician repeats steps 7-9 for each car

SUC-4 System Testing

SUC-4	System Testing
Branch A	Branch from step S (S=2 or 7) of the MSS: Error on a floor/car SA1. The technician fixes the error (SUC-5) SA2. The technician repeats the failed test and checks that it is nominal SA3. Return to MSS.
Branch B	Exception from step SA1 of Branch A: The technician can not fix the error SA1B1. The technician marks the failure in the report SA1B2. Return to steps 5 or 9 in the MSS (as appropriate)
Requirements trackback	…

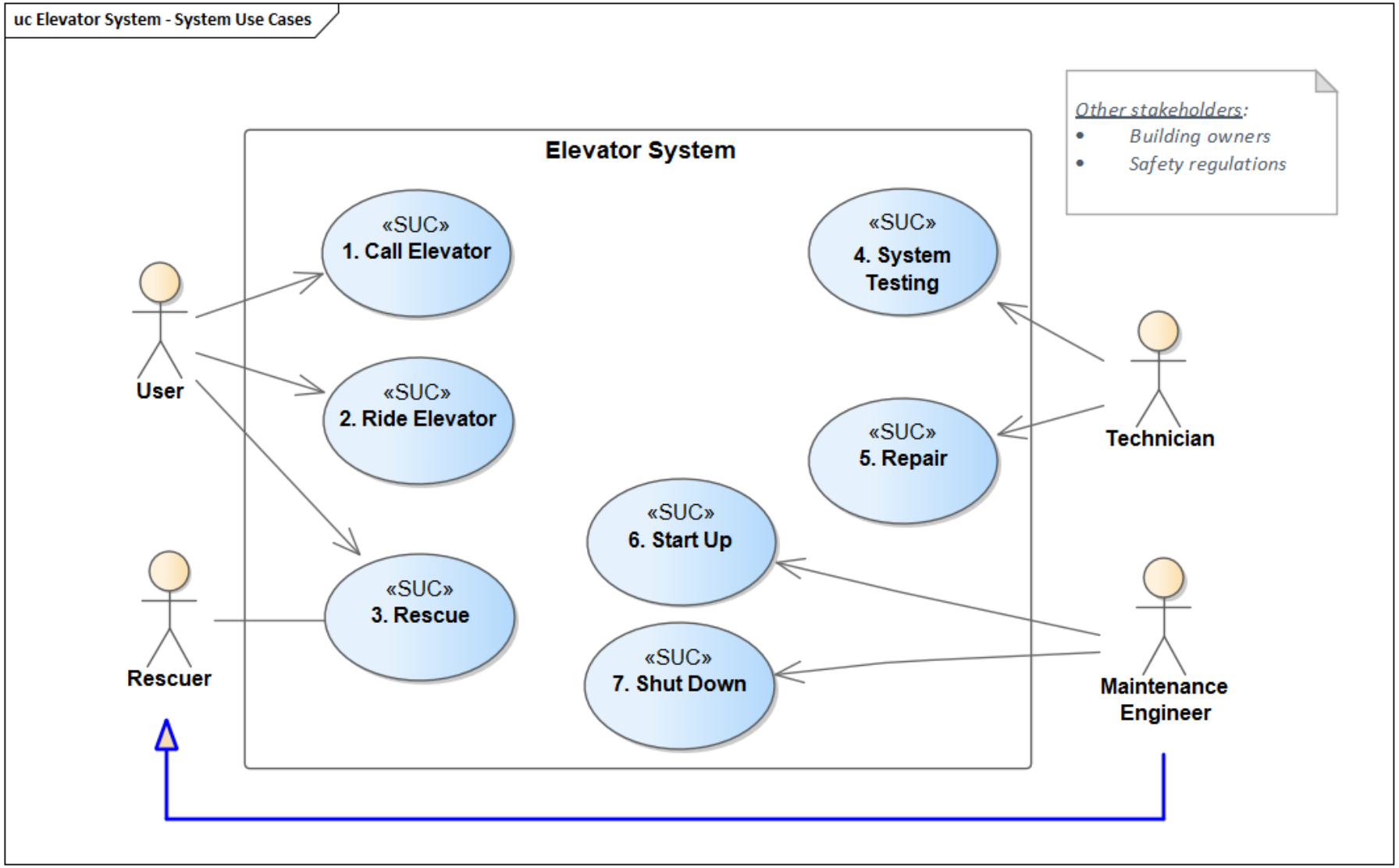
SUC-6 Start Up

SUC-6	Start Up
Actors and Goals	Maintenance Engineer: Put the system in operational mode
Stakeholders and Interests	Passengers: The system is ready to use
Pre-conditions	• The system is not operational
Post-conditions	• The system is operational and ready to use
Trigger	**Maintenance Engineer** turns on the system
Main Success Sequence (MSS)	1. The system initializes all its components (cars, panels, etc.) 2. The system checks that all of its components are working 3. The system gives a “all ok” indication
Branch A	Exception from Step 2 of the MSS: Not all system components are in proper working order 2A1. The system gives a “not ok” indication 2A2. End of use case
Requirements Trackback	…

Requirements trackback to the use case model

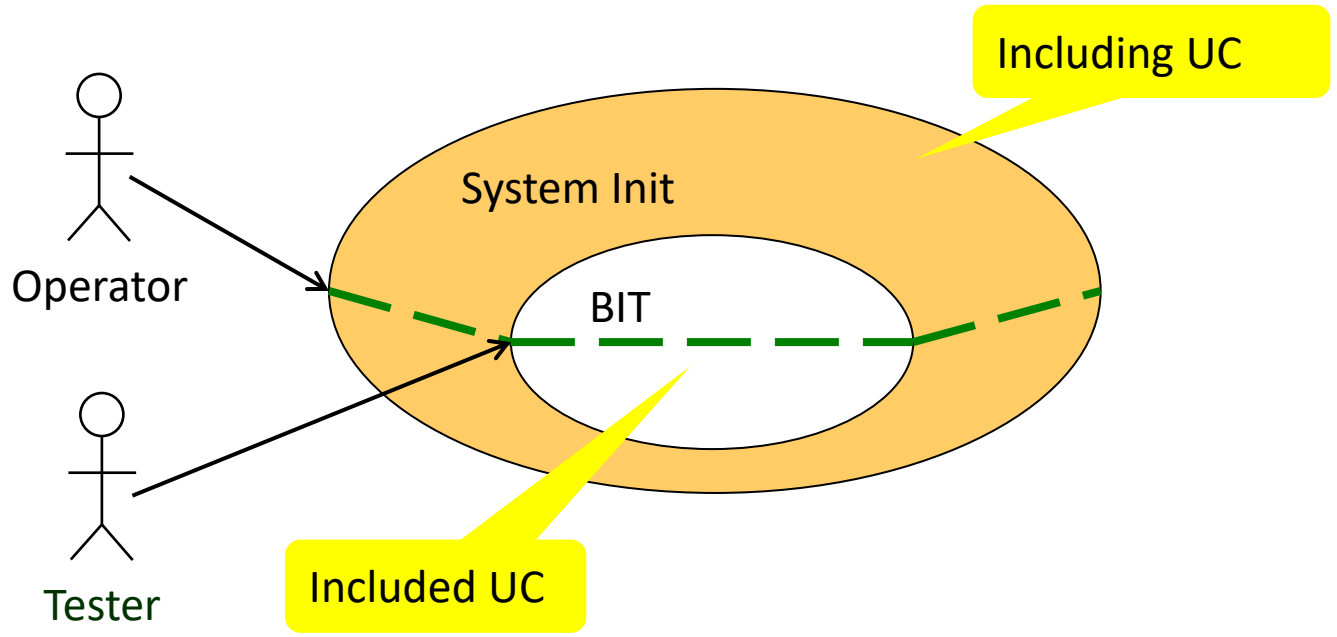
ID	Requirement	Type	SUC
	…		
3	A passenger who is on a floor … presses on a button for the direction he wants to travel	OR	SUC-1
4	The direction button on the floor lights up when pressed if not already lit	OR	SUC-1
5	After pressing a floor's button, a car traveling in the desired direction will arrive	OR	SUC-1
6	When a car arrives, the door opens, and the floor's button turns off	OR	SUC-1
	…		
20	A passenger in car who wants to ride to a floor presses the appropriate button	OR	SUC-2
21	The floor button lights up when pressed if not already lit	OR	SUC-2
22	When the car arrives at a floor, the door opens, and the floor's button turns off	OR	SUC-2
23	If car gets stuck while in motion, the passenger can call for help	OR	SUC-3
24	…The system is tested … by a licensed technician	OR	SUC-4
25	If the technician finds a problem, he tries to repair it…	OR	SUC-5

Basic Elevator Use Case Diagram

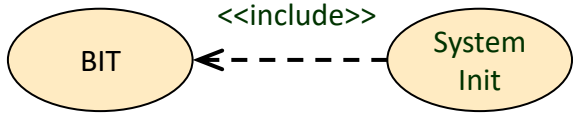


“Includes” Dependency between use cases

- Use Case A includes/contains Use Case B if B is an integral part of A
  - B can also run independently via other actors (e.g. Built in Test)



In UML



When to use <<include>>?

- Break a use case into pieces and create an included use case if:

The included use case is very long (has many steps)

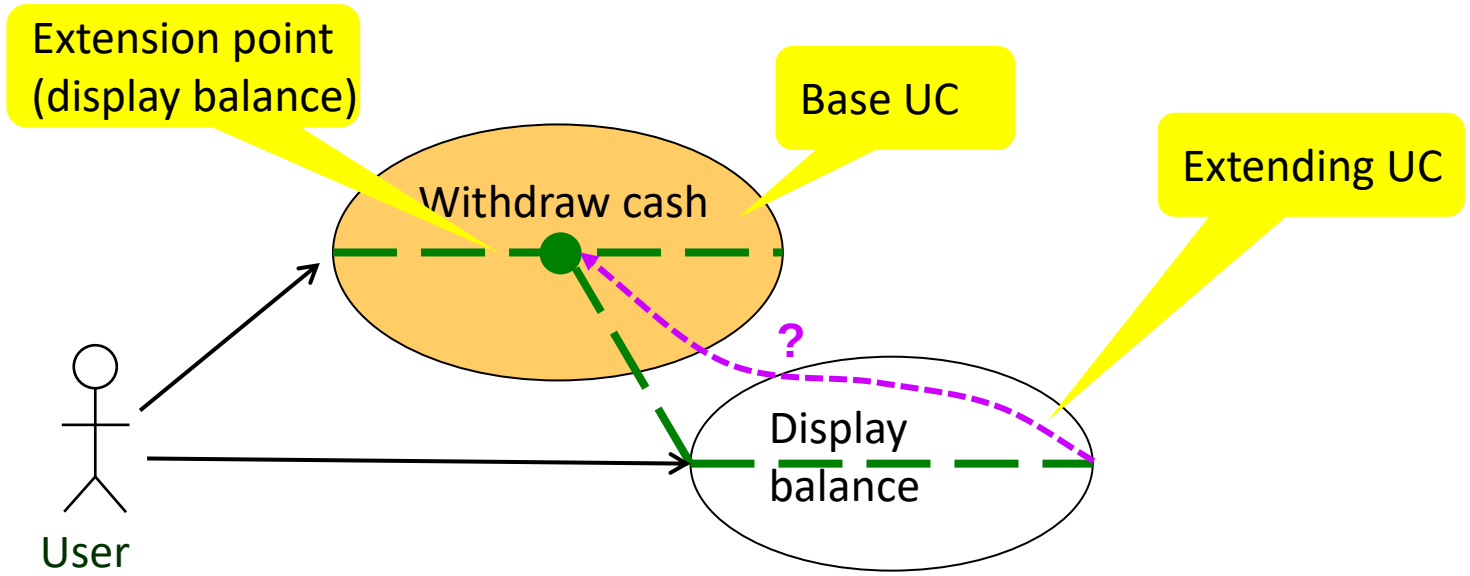
The included activity is shared by more than one use case

- Then it's <<include>>-ed by all of them

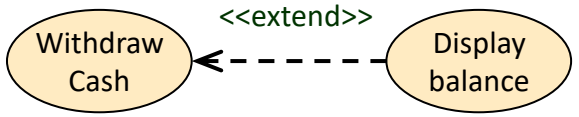
The included use case can be performed independently

“Extends” dependency between use cases

- Use Case B extends Use Case A if B might be performed as part of A
  - B is performed at an extension point within A
  - B can also be run independently by other actors (e.g. display balance while withdrawing cash)



In UML



When to use <<extend>>?

- In most cases, you can achieve the same thing by using branches within the base use case
- It makes sense to divide a use case into a base + extension when:

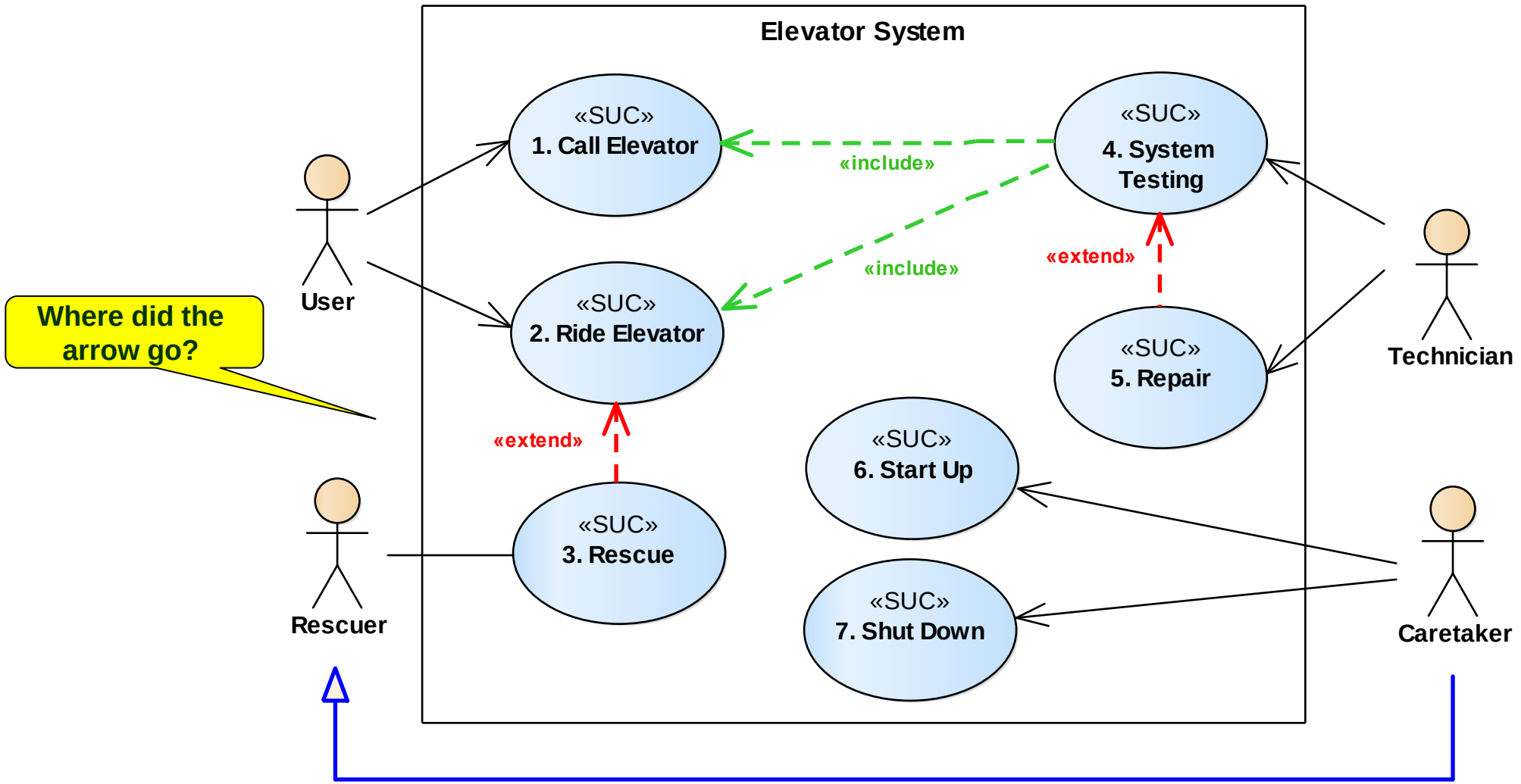
The extension steps are very long (many steps)

The extension steps are shared by more than one use case

- Then it <<extend>>-s all of them

The extension steps can be performed independently

Elevator Use Case Diagram with dependencies



In Class Assignment

- Write the Use Case details for the following ePark use cases
  - Register and Check In
  - Enter park attraction
  - Monitor attraction
- Use the format in the slides for the Use Case details
- Change or improve your Use Case Diagram
  - Add <<include>> and <<extends>> as appropriate

Conclusion

- Use cases
  - –Identifying system use cases
  - –Use Case Definition and diagram
  - –Specifying use case contents