

COMP 1531

Software Engineering Fundamentals

Week 03

Requirements Engineering

Use-Case Modelling

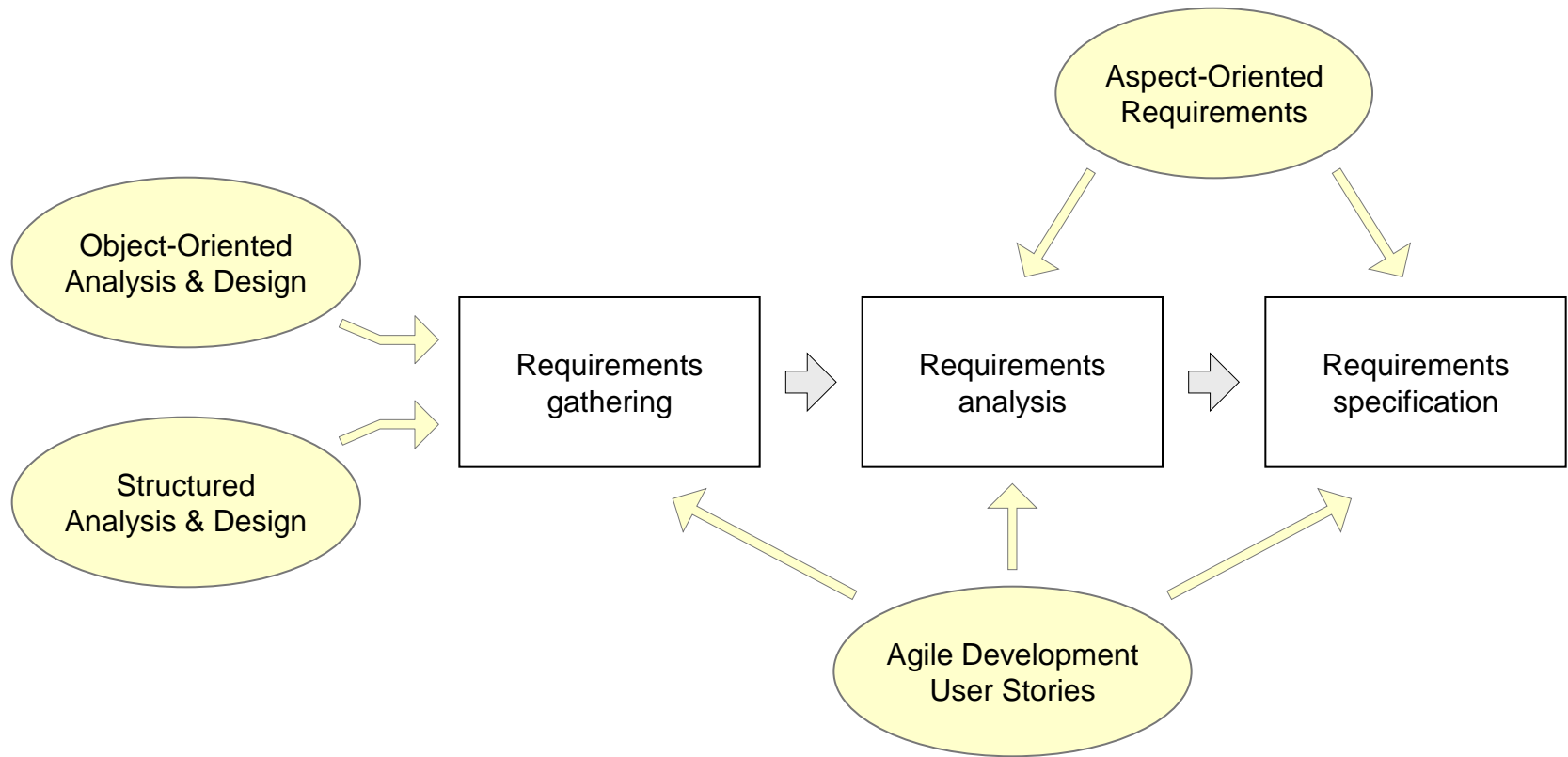
Aarthi Natarajan

(© Slides from Software Engineering, Ivan Marsic, 2012)

Topics

- Requirements Engineering Components
- Requirements and User Stories
- Types of Requirements
- Effort Estimation (Agile Methods)
- Use Case Modelling

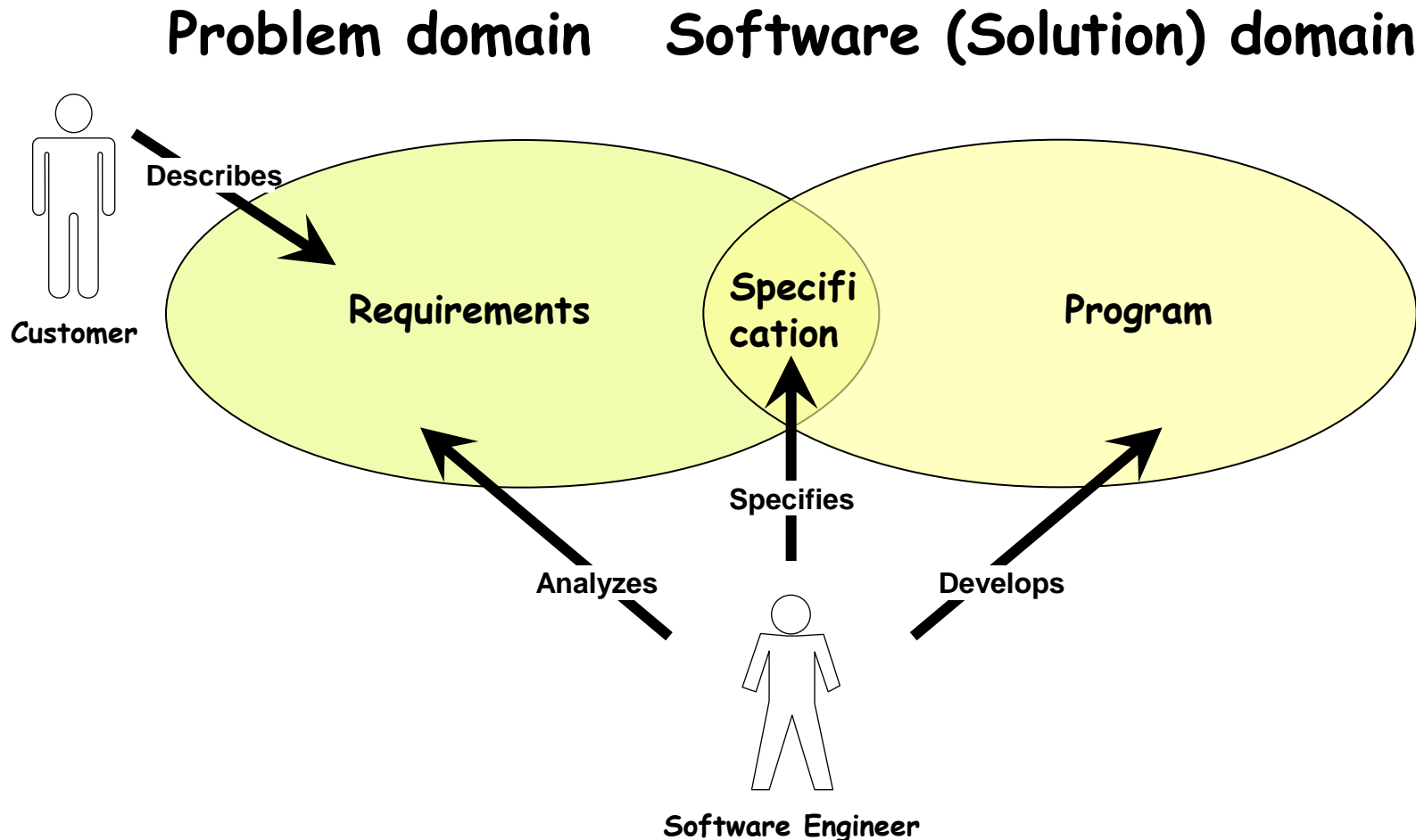
Requirements Process



Requirements Engineering Components

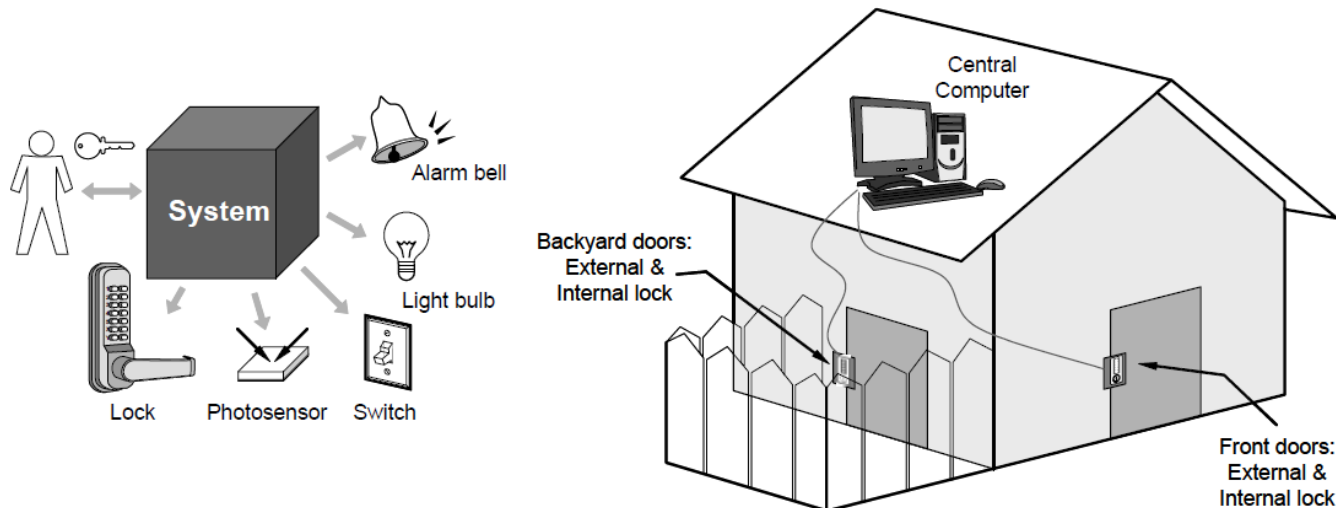
- Requirements gathering
 - (a.k.a. “requirements elicitation”) helps the customer to define what is required: what is to be accomplished, how the system will fit into the needs of the business, and how the system will be used on a day-to-day basis
- Requirements analysis
 - refining and modifying the gathered requirements
- Requirements specification
 - documenting the system requirements in a semiformal or formal manner to ensure clarity, consistency, and completeness

Requirements and Specification



Home Access Case Study

- A home access control system for several functions such as door lock control, lighting control, intrusion detection
- First iteration - Support basic door unlocking and locking functions

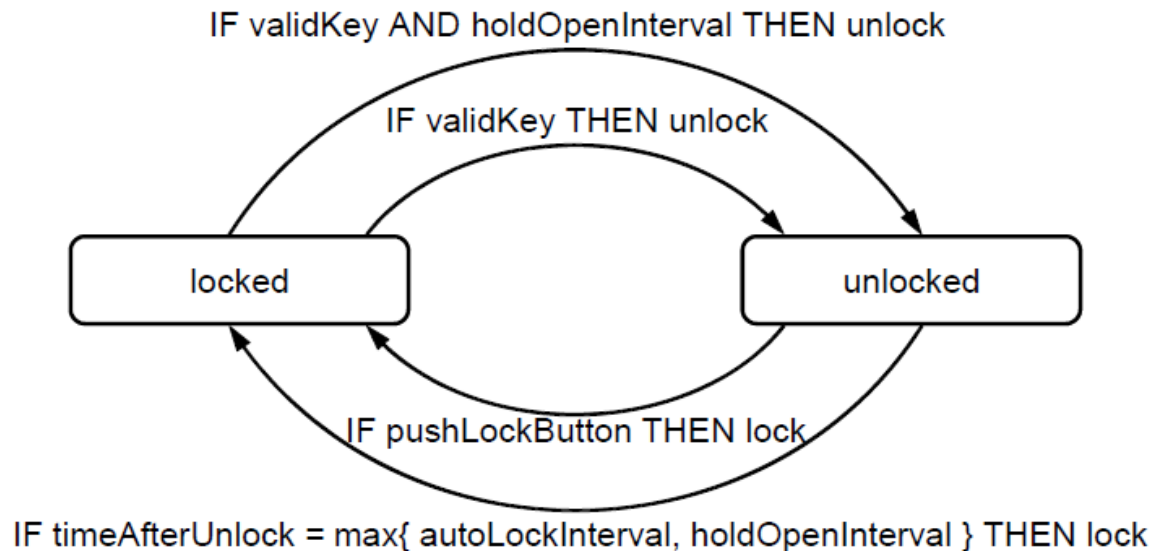


Requirements Analysis Challenges

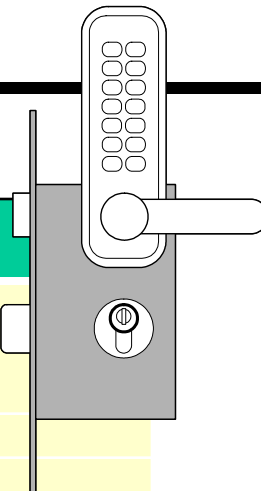
- User-identification: Several choices
 - What you carry on you (physical key or another gadget)
 - What you know (password)
 - Who you are (biometric feature, such as fingerprint, voice, face, or iris)
- With user-constraints: user should not need to carry any gadgets for identification; and, the identification mechanism should be cheap.
 - rules out a door-mounted reader for magnetic strip ID cards or RFID tags, biometric identification mechanisms
- Solution
 - simple authentication based on a valid key (memorised by user)
 - Anyone with knowledge of key permitted to enter (no true authentication)

Requirements Analysis Challenges

- But the problem is still complex
 - Handle failed attempts ?
 - Accommodate forgetful users - autoLock after #Interval
 - Or perhaps keep door open longer - holdOpenInterval



Example System Requirements

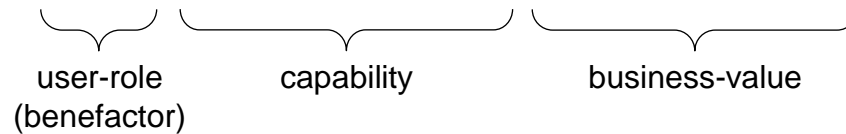


Identifier	Priority	Requirement
REQ1	5	The system shall keep the door locked at all times, unless commanded otherwise by authorized user. When the lock is disarmed, a countdown shall be initiated at the end of which the lock shall be automatically armed (if still disarmed).
REQ2	2	The system shall lock the door when commanded by pressing a dedicated button.
REQ3	5	The system shall, given a valid key code, unlock the door and activate other devices.
REQ4	4	The system should allow mistakes while entering the key code. However, to resist “dictionary attacks,” the number of allowed failed attempts shall be small, say three, after which the system will block and the alarm bell shall be sounded.
REQ5	2	The system shall maintain a history log of all attempted accesses for later review.
REQ6	2	The system should allow adding new authorized persons at runtime or removing existing ones.
REQ7	2	The system shall allow configuring the preferences for device activation when the user provides a valid key code, as well as when a burglary attempt is detected.
REQ8	1	The system should allow searching the history log by specifying one or more of these parameters: the time frame, the actor role, the door location, or the event type (unlock, lock, power failure, etc.). This function shall be available over the Web by pointing a browser to a specified URL.
REQ9	1	The system should allow filing inquiries about “suspicious” accesses. This function shall be available over the Web.

- Problem: Requirements prioritization.
- See how solved in **agile methods**.

User Stories For Home Access Control

As a tenant, I can unlock the doors to enter my apartment.



- Similar to system requirements, but focus on the user benefits, instead on system features.
- Preferred tool in **agile methods**.

Example User Stories

Identifier	User Story	Size
ST-1	As an authorized person (tenant or landlord), I can keep the doors locked at all times.	4 points
ST-2	As an authorized person (tenant or landlord), I can lock the doors on demand.	3 pts
ST-3	The lock should be automatically locked after a defined period of time.	6 pts
ST-4	As an authorized person (tenant or landlord), I can unlock the doors. (Test: Allow a small number of mistakes, say three.)	9 points
ST-5	As a landlord, I can at runtime manage authorized persons.	10 pts
ST-6	As an authorized person (tenant or landlord), I can view past accesses.	6 pts
ST-7	As a tenant, I can configure the preferences for activation of various devices.	6 pts
ST-8	As a tenant, I can file complaint about “suspicious” accesses.	6 pts

- Story priority is given by its order of appearance on the work backlog (described next)
- Size (Effort) estimated in story points (last column)

Types of Requirements

- Functional Requirements
- Non-functional requirements (or quality requirements)
 - FURPS+
 - Functionality (security), Usability, Reliability, Performance , Supportability
- On-screen appearance requirements

On-screen Appearance Requirements

- Do not waste your time and your customer's time by creating elaborate screen shots with many embellishments, coloring, shading, etc., that serves only to distract attention from most important aspects of the interface
- Hand-drawing the proposed interface forces you to *economize* and focus on the most important features
- Only when there is a consensus that a good design is reached, invest effort to prototype the interface

Tools for Requirements Eng.

- Tools, such as user stories and use cases, used for:
 - Determining what exactly the user needs (“requirements analysis”)
 - Writing a description of what system will do (“requirements specification”)
- Difficult to use the same tool for different tasks (analysis vs. specification)

Acceptance Tests

- Means of assessing that the requirements are met as expected
- Conducted by the customer
- An acceptance test describes whether the system will pass or fail the test, given specific input values
- Cannot ever guarantee 100% coverage of all usage scenarios, but *systematic approach* can increase the degree of coverage

Project Estimation using User Story Points

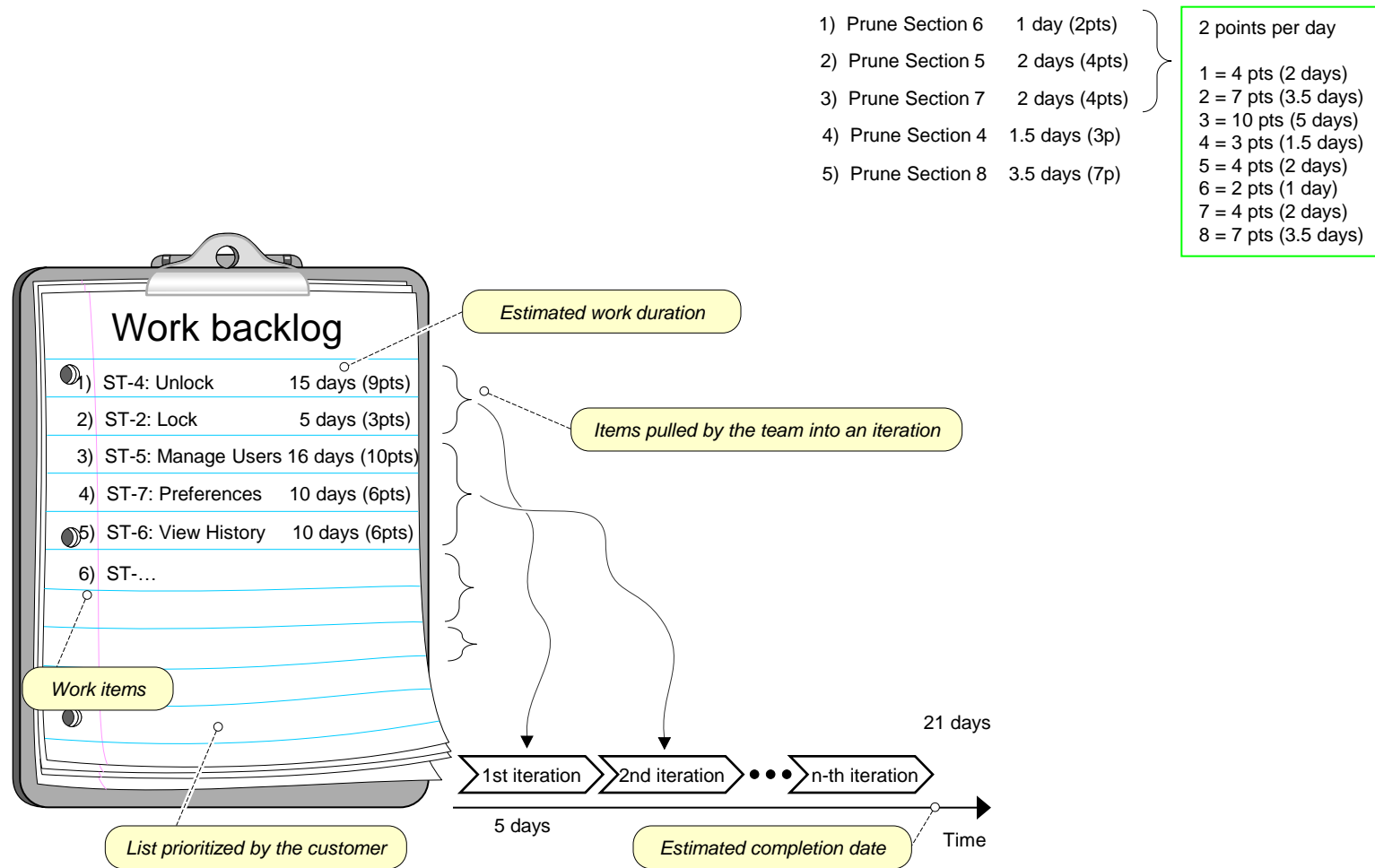
- Size points assigned to each user story
- Total work size estimate:
 - Total size = $\sum (\text{points-for-story } i), \quad i = 1..N$
- **Velocity** (= team's productivity)
 - estimated from experience (Number of user-story points that the team can complete per single iteration)
- Estimate the work duration

$$\text{Project duration} = \frac{\text{Path size}}{\text{Travel velocity}}$$

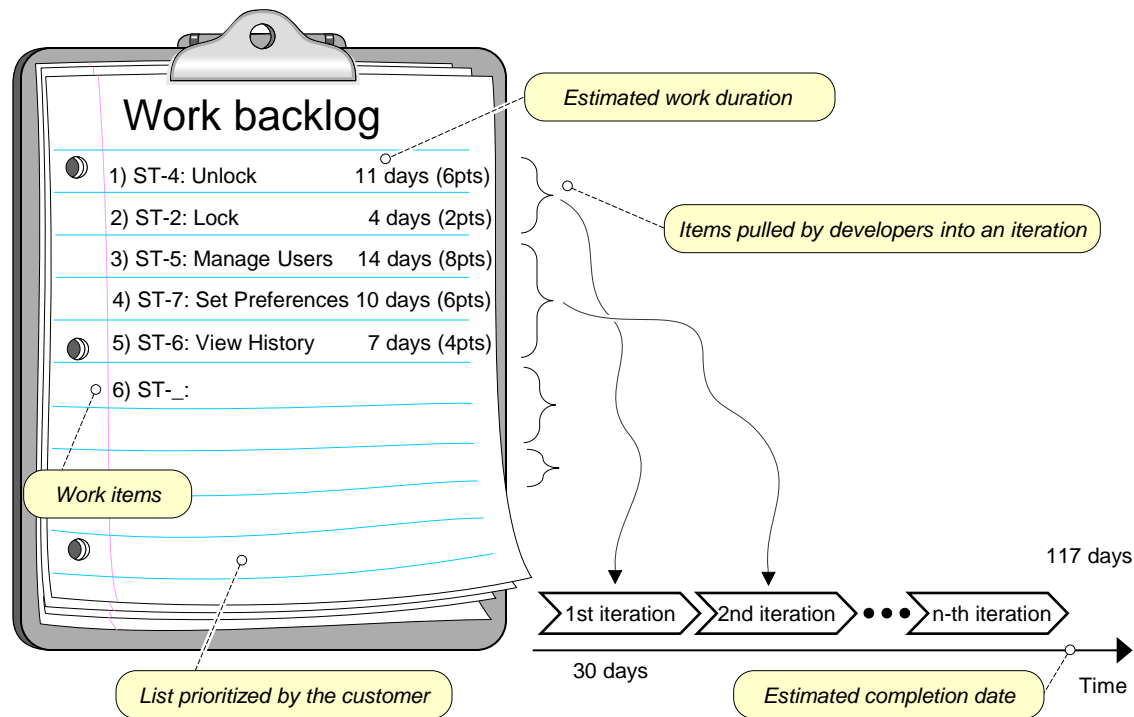
Example User Stories

Identifier	User Story	Size
ST-1	As an authorized person (tenant or landlord), I can keep the doors locked at all times.	4 points
ST-2	As an authorized person (tenant or landlord), I can lock the doors on demand.	3 pts
ST-3	The lock should be automatically locked after a defined period of time.	6 pts
ST-4	As an authorized person (tenant or landlord), I can unlock the doors. (Test: Allow a small number of mistakes, say three.)	9 points
ST-5	As a landlord, I can at runtime manage authorized persons.	10 pts
ST-6	As an authorized person (tenant or landlord), I can view past accesses.	6 pts
ST-7	As a tenant, I can configure the preferences for activation of various devices.	6 pts
ST-8	As a tenant, I can file complaint about “suspicious” accesses.	6 pts

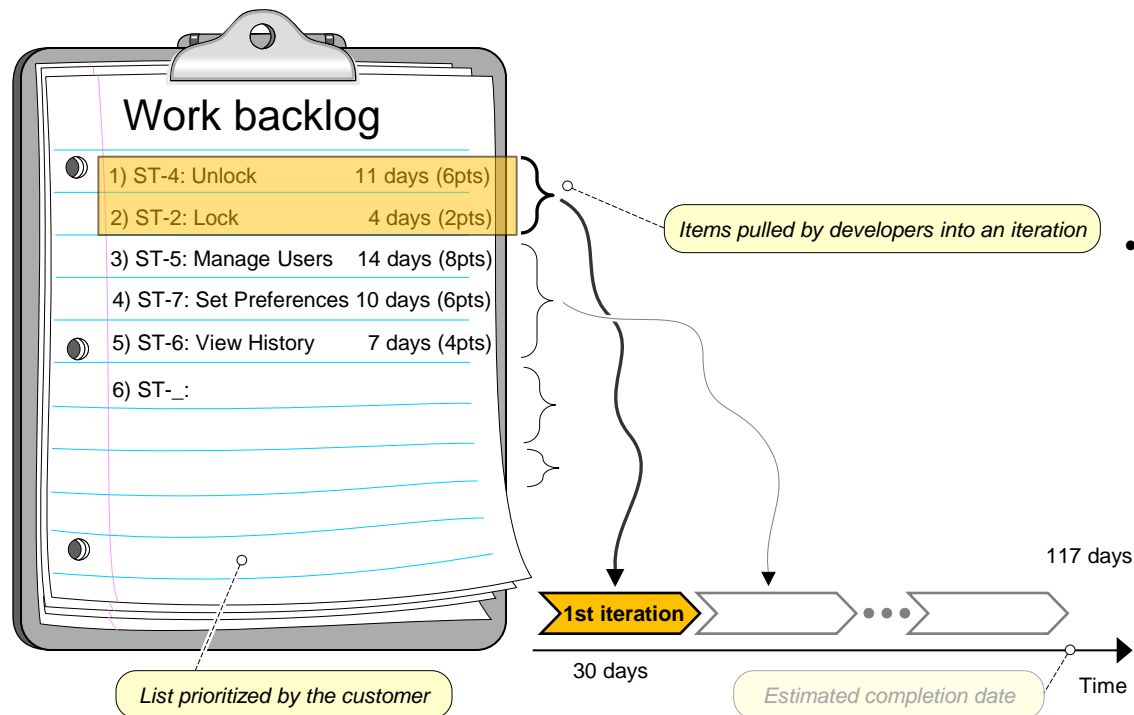
Agile Estimation of Project Effort



Agile Estimation of Project Effort

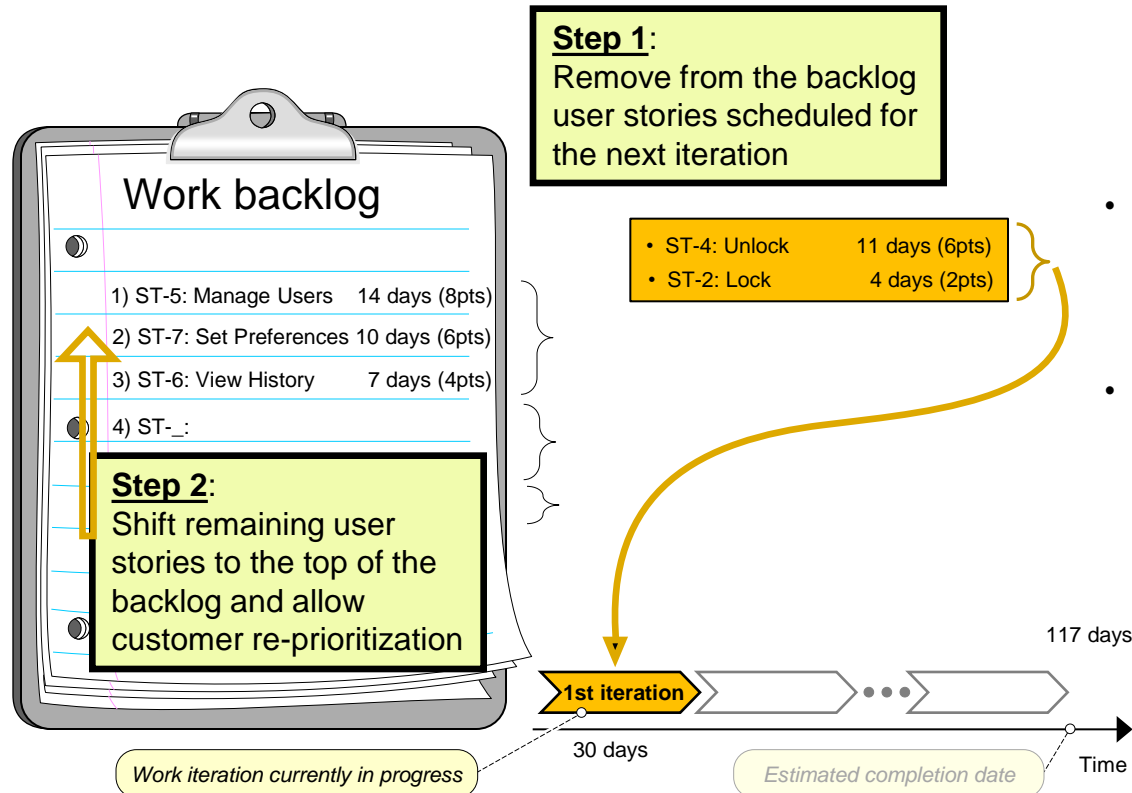


Agile Prioritization of Work



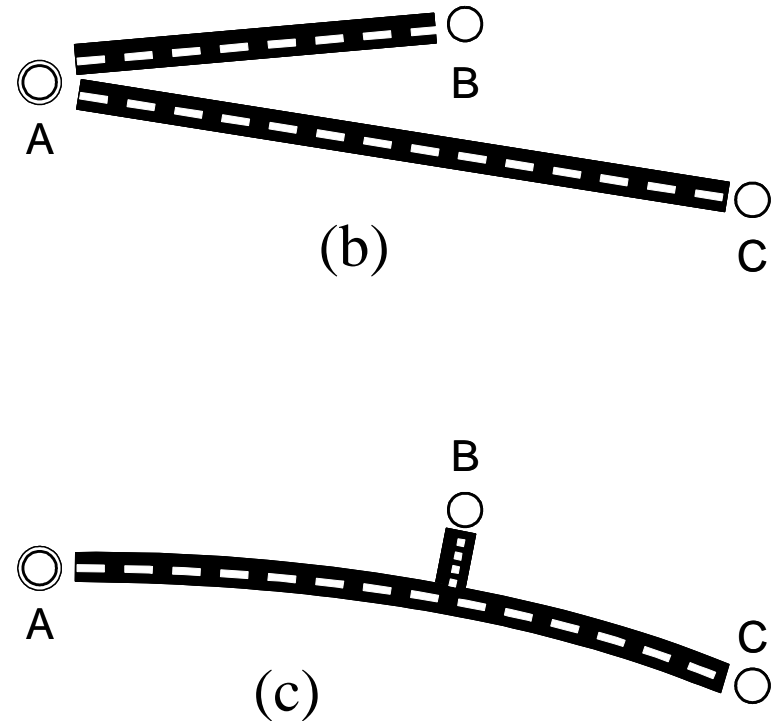
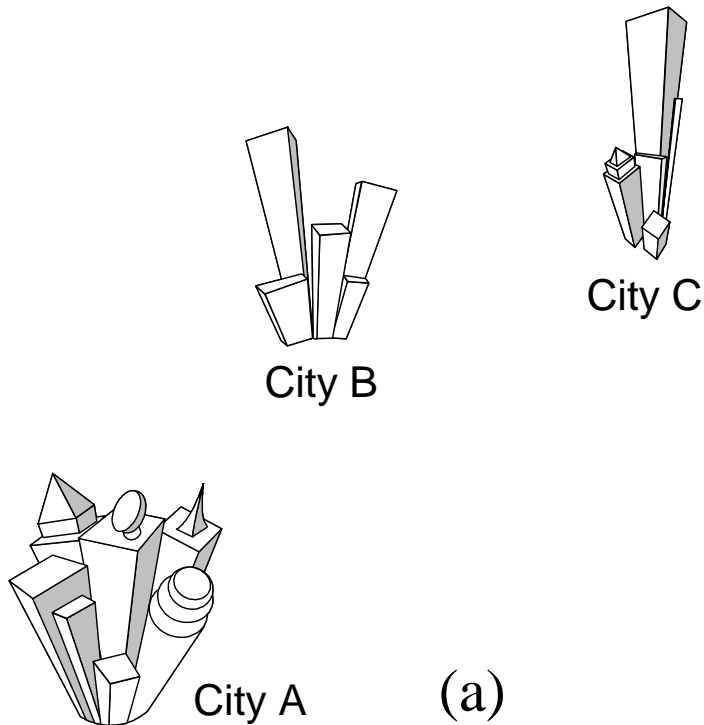
- Instead of assigning priorities, the customer creates an ordered list of user stories
- Developers simply remove the top list items and work on them in the next iteration

Tradeoff between Customer Flexibility and Developer Stability



- Items pulled by developers into an iteration are not subject to further customer prioritization
- Developers have a **steady goal** until the end of the current iteration
- Customer has **flexibility** to change priorities in response to changing market forces

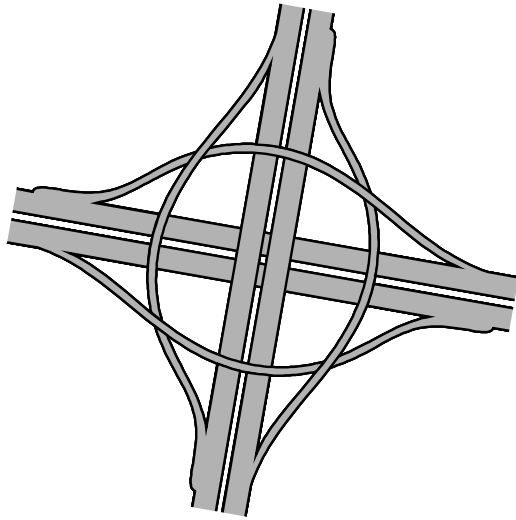
How To Combine the Part Sizes?



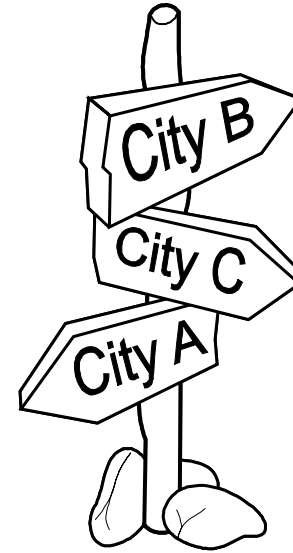
Costs are not always additive

But, solution (c) is not necessarily “cheaper” than (b) ...

Additional Costs



Highway traffic-circle interchange



Traffic signs

Topics

- Actors, Goals
- Sketchy/Summary Use Cases
- Use Case Diagram
- Traceability Matrix
- System Boundary and Subsystems
- Detailed Use Case Specification
- System Sequence Diagrams
- Security and Risk Management

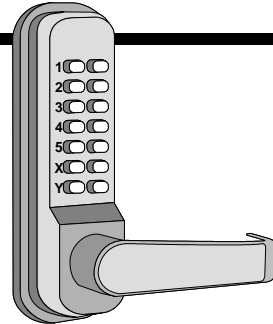
Use Case Modelling

Use Cases

- Used for Functional Requirements Analysis and Specification
- A *use case* is a step-by-step description of how a user will use the system-to-be to accomplish business goals
 - Detailed use cases are usually written as *usage scenarios* or *scripts*, showing an envisioned sequence of actions and interactions between the external actors and the system-to-be

Deriving Use Cases from System Requirements

REQ1: Keep door locked and auto-lock
 REQ2: Lock when "LOCK" pressed
 REQ3: Unlock when valid key provided
 REQ4: Allow mistakes but prevent dictionary attacks
 REQ5: Maintain a history log
 REQ6: Adding/removing users at runtime
 REQ7: Configuring the device activation preferences
 REQ8: Inspecting the access history
 REQ9: Filing inquiries



Actor	Actor's Goal (what the actor intends to accomplish)	Use Case Name
Landlord	To disarm the lock and enter, and get space lighted up.	Unlock (UC-1)
Landlord	To lock the door & shut the lights (sometimes?).	Lock (UC-2)
Landlord	To create a new user account and allow access to home.	AddUser (UC-3)
Landlord	To retire an existing user account and disable access.	RemoveUser (UC-4)
Tenant	To find out who accessed the home in a given interval of time and potentially file complaints.	InspectAccessHistory (UC-5)
Tenant	To disarm the lock and enter, and get space lighted up.	Unlock (UC-1)
Tenant	To lock the door & shut the lights (sometimes?).	Lock (UC-2)
Tenant	To configure the device activation preferences.	SetDevicePrefs (UC-6)
LockDevice	To control the physical lock mechanism.	UC-1, UC-2
LightSwitch	To control the lightbulb.	UC-1, UC-2
[to be identified]	To auto-lock the door if it is left unlocked for a given interval of time.	AutoLock (UC-2)

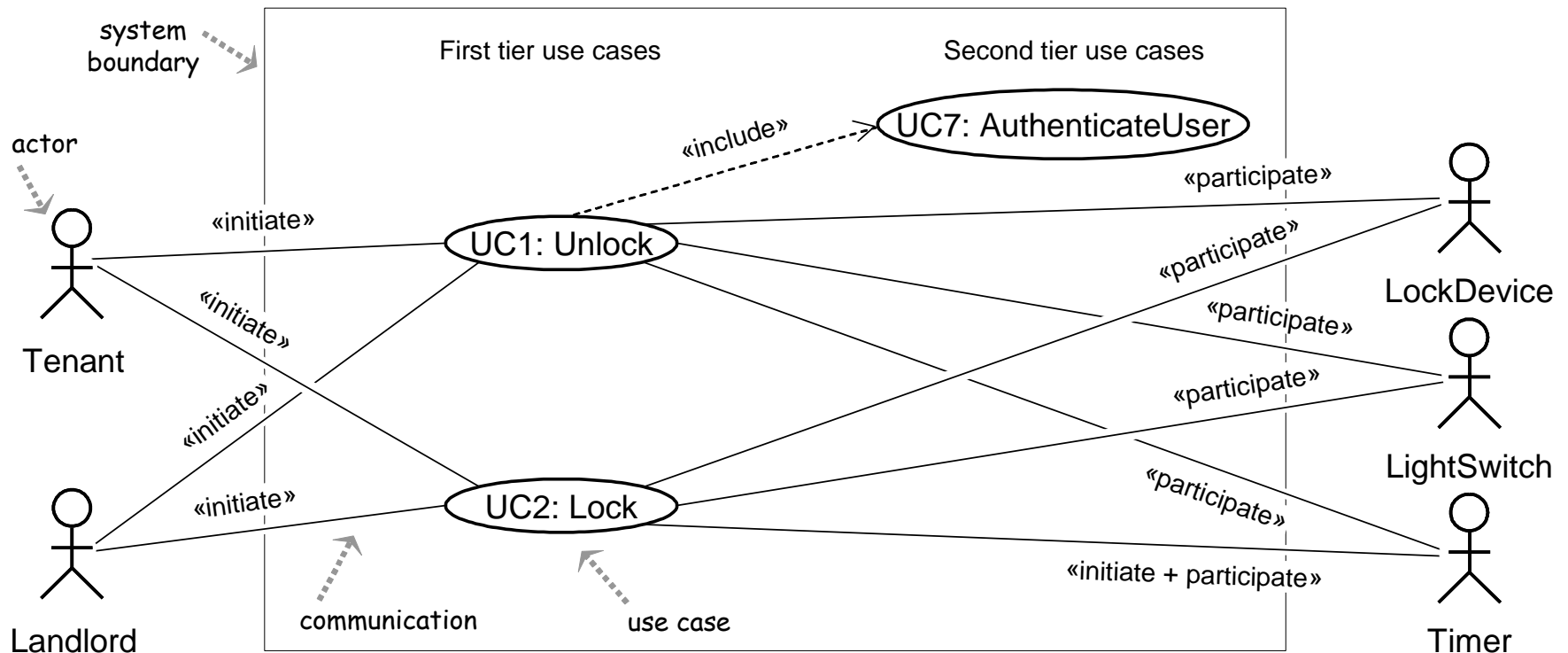
(Actors already given if working from user stories instead of system requirements)

Types of Actors

- ***Initiating actor*** (also called *primary actor* or simply "user"): initiates the use case to achieve a goal
- ***Participating actor*** (also called *secondary actor*): participates in the use case but does not initiate it. Subtypes of participating actors:
 - ***Supporting actor***: helps the system-to-be to complete the use case
 - ***Offstage actor***: passively participates in the use case, i.e., neither initiates nor helps complete the use case, but may be notified about some aspect of it (e.g., for keeping records)

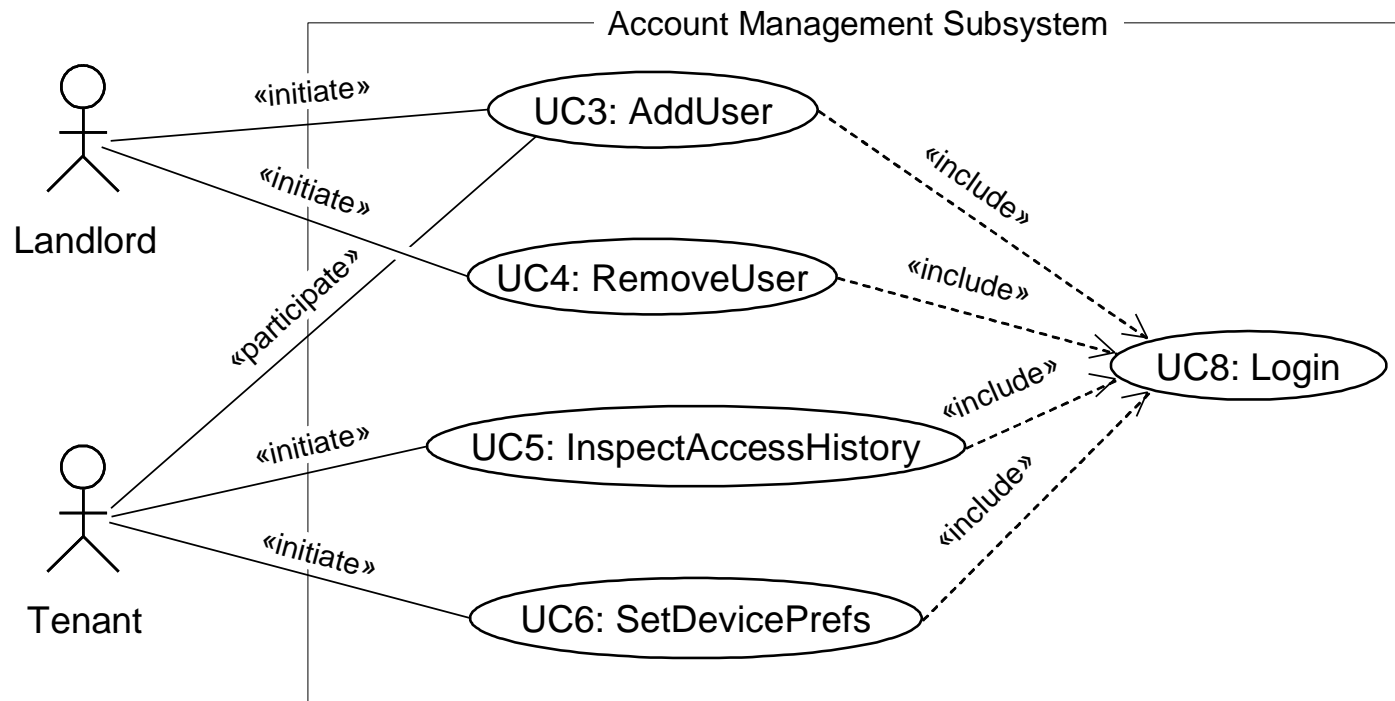
Use Case Diagram: Device Control

UC1: Unlock
UC2: Lock
UC3: AddUser
UC4: RemoveUser
UC5: InspectAccessHistory
UC6: SetDevicePrefs
UC7: AuthenticateUser
UC8: Login

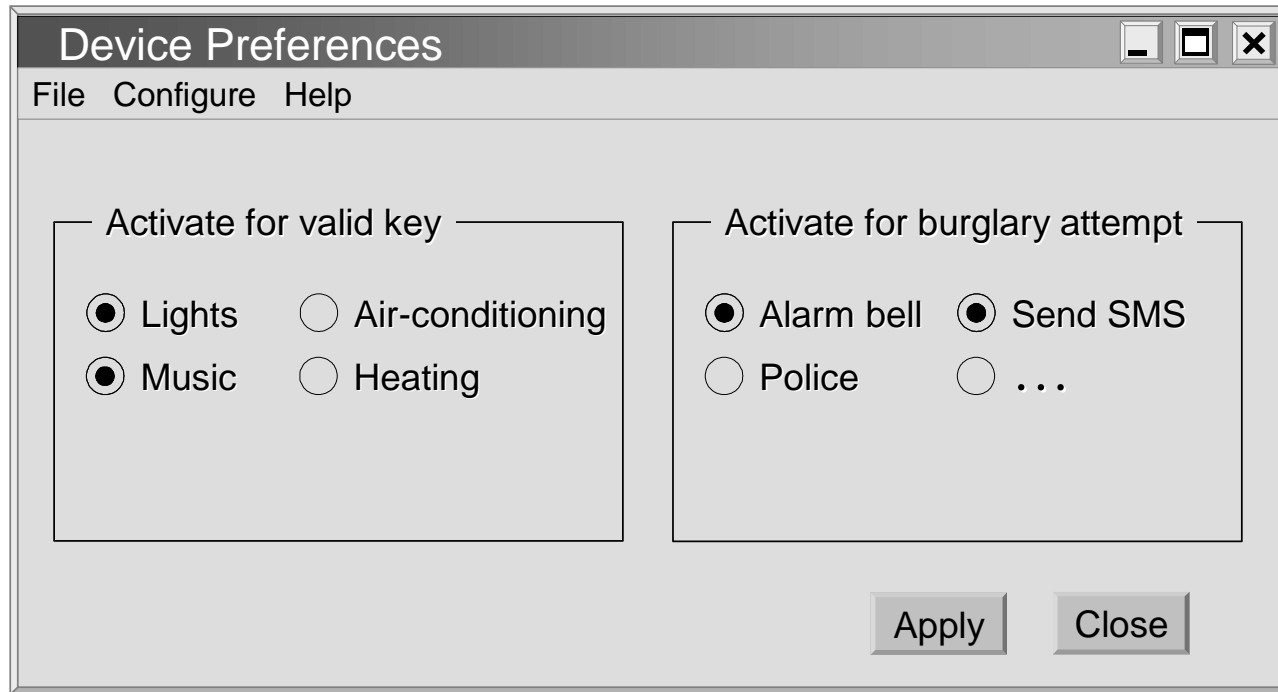


Use Case Diagram: Account Management

UC1: Unlock
UC2: Lock
UC3: AddUser
UC4: RemoveUser
UC5: InspectAccessHistory
UC6: SetDevicePrefs
UC7: AuthenticateUser
UC8: Login



GUI for UC6: Set Device Pref's

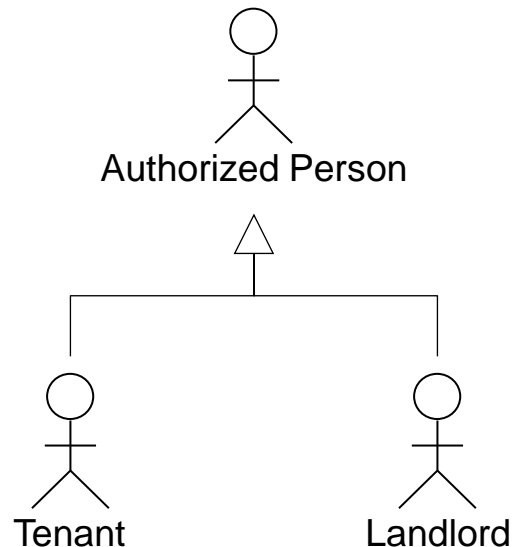


(NOTE: Lock device is mandatory, not an option)

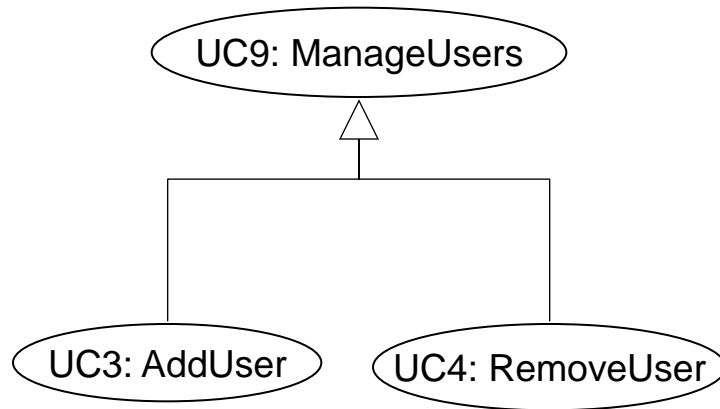
Use Case Generalizations

- More abstract representations can be derived from particular representations

UC1: Unlock
UC2: Lock
UC3: AddUser
UC4: RemoveUser
UC5: InspectAccessHistory
UC6: SetDevicePrefs
UC7: AuthenticateUser
UC8: Login



Actor Generalization

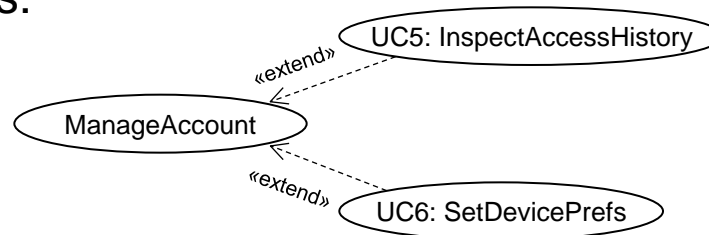


Use Case Generalization

Optional Use Cases: «extend»

UC1: Unlock
UC2: Lock
UC3: AddUser
UC4: RemoveUser
UC5: InspectAccessHistory
UC6: SetDevicePrefs
UC7: AuthenticateUser
UC8: Login

Example optional use cases:

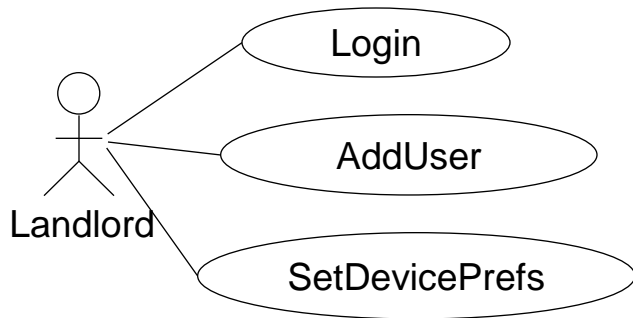


Key differences between «include» and «extend» relationships

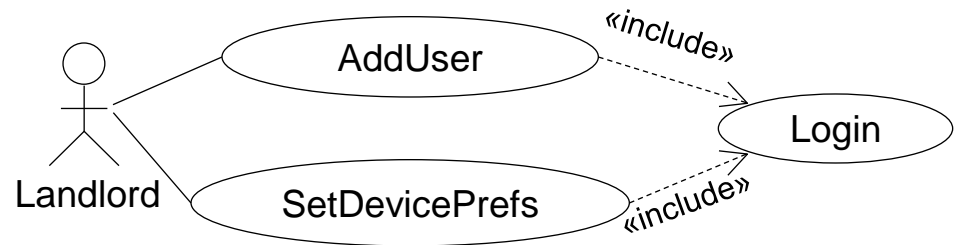
	Included use case	Extending use case
Is this use case optional?	No	Yes
Is the base use case complete without this use case?	No	Yes
Is the execution of this use case conditional?	No	Yes
Does this use case change the behavior of the base use case?	No	Yes

Login Use Case?

BAD:



GOOD:



Traceability Matrix (1)

Mapping: System requirements to Use cases

REQ1: Keep door locked and auto-lock
 REQ2: Lock when "LOCK" pressed
 REQ3: Unlock when valid key provided
 REQ4: Allow mistakes but prevent dictionary attacks
 REQ5: Maintain a history log
 REQ6: Adding/removing users at runtime
 REQ7: Configuring the device activation preferences
 REQ8: Inspecting the access history
 REQ9: Filing inquiries

UC1: Unlock
 UC2: Lock
 UC3: AddUser
 UC4: RemoveUser
 UC5: InspectAccessHistory
 UC6: SetDevicePrefs
 UC7: AuthenticateUser
 UC8: Login

Req't	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
REQ1	5	X	X						
REQ2	2		X						
REQ3	5	X						X	
REQ4	4	X						X	
REQ5	2	X	X						
REQ6	1			X	X				X
REQ7	2						X		X
REQ8	1					X			X
REQ9	1					X			X
Max PW		5	2	2	2	1	5	2	1
Total PW		15	3	2	2	3	9	2	3

Continued for domain model, design diagrams, ...

Traceability Matrix Purpose

- To check that all requirements are covered by the use cases
- To check that none of the use cases is introduced without a reason (i.e., created not in response to any requirement)
- To prioritize the work on use cases

Schema for Detailed Use Cases

Use Case UC-#:	Name / Identifier [verb phrase]	
Related Requirements:	List of the requirements that are addressed by this use case	
Initiating Actor:	Actor who initiates interaction with the system to accomplish a goal	
Actor's Goal:	Informal description of the initiating actor's goal	
Participating Actors:	Actors that will help achieve the goal or need to know about the outcome	
Preconditions:	What is assumed about the state of the system before the interaction starts	
Postconditions:	What are the results after the goal is achieved or abandoned; i.e., what must be true about the system at the time the execution of this use case is completed	
Flow of Events for Main Success Scenario:		
→	1.	The initiating actor delivers an action or stimulus to the system (the arrow indicates the direction of interaction, to- or from the system)
←	2.	The system's reaction or response to the stimulus; the system can also send a message to a participating actor, if any
→	3.	...
Flow of Events for Extensions (Alternate Scenarios):		
What could go wrong? List the exceptions to the routine and describe how they are handled		
→	1a.	For example, actor enters invalid data
←	2a.	For example, power outage, network failure, or requested data unavailable
		...
The arrows on the left indicate the direction of interaction: → Actor's action; ← System's reaction		

Use Case 1: Unlock

Use Case UC-1: Unlock

Related Requirem'ts: REQ1, REQ3, REQ4, and REQ5 stated in Table 2-1

Initiating Actor: Any of: Tenant, Landlord

Actor's Goal: To disarm the lock and enter, and get space lighted up automatically.

Participating Actors: LockDevice, LightSwitch, Timer

Preconditions:

- The set of valid keys stored in the system database is non-empty.
- The system displays the menu of available functions; at the door keypad the menu choices are "Lock" and "Unlock."

Postconditions: The auto-lock timer has started countdown from autoLockInterval.

Flow of Events for Main Success Scenario:

- 1. **Tenant/Landlord** arrives at the door and selects the menu item "Unlock"
2. include::AuthenticateUser (UC-7)
- ← 3. **System** (a) signals to the **Tenant/Landlord** the lock status, e.g., "disarmed," (b) signals to **LockDevice** to disarm the lock, and (c) signals to **LightSwitch** to turn the light on
- ← 4. **System** signals to the **Timer** to start the auto-lock timer countdown
- 5. **Tenant/Landlord** opens the door, enters the home [and shuts the door and locks]

Subroutine «include» Use Case

Use Case UC-7: **AuthenticateUser** (sub-use case)

Related Requirements:	REQ3, REQ4 stated in Table 2-1
Initiating Actor:	Any of: Tenant, Landlord
Actor's Goal:	To be positively identified by the system (at the door interface).
Participating Actors:	AlarmBell, Police
Preconditions:	<ul style="list-style-type: none">• The set of valid keys stored in the system database is non-empty.• The counter of authentication attempts equals zero.
Postconditions:	None worth mentioning.

Flow of Events for Main Success Scenario:

- ← 1. **System** prompts the actor for identification, e.g., alphanumeric key
- 2. **Tenant/Landlord** supplies a valid identification key
- ← 3. **System** (a) verifies that the key is valid, and (b) signals to the actor the key validity

Flow of Events for Extensions (Alternate Scenarios):

2a. **Tenant/Landlord** enters an invalid identification key

- ← 1. **System** (a) detects error, (b) marks a failed attempt, and (c) signals to the actor
System (a) detects that the count of failed attempts exceeds the maximum allowed
- ← 1a. number, (b) signals to sound **AlarmBell**, and (c) notifies the **Police** actor of a possible break-in
- 2. **Tenant/Landlord** supplies a valid identification key
- 3. Same as in Step 3 above

Acceptance Test Case for UC-7 Authenticate User

Test-case Identifier: TC-1	
Use Case Tested: UC-1, main success scenario, and UC-7	
Pass/fail Criteria: The test passes if the user enters a key that is contained in the database, with less than a maximum allowed number of unsuccessful attempts	
Input Data: Numeric keycode, door identifier	
Test Procedure:	Expected Result:
Step 1. Type in an incorrect keycode and a valid door identifier	System beeps to indicate failure; records unsuccessful attempt in the database; prompts the user to try again
Step 2. Type in the correct keycode and door identifier	System flashes a green light to indicate success; records successful access in the database; disarms the lock device

Use Case 2: Lock

Use Case UC-2: Lock

Related Requirements: REQ1, REQ2, and REQ5 stated in Table 2-1

Initiating Actor: Any of: Tenant, Landlord, or Timer

Actor's Goal: To lock the door & get the lights shut automatically (?)

Participating Actors: LockDevice, LightSwitch, Timer

Preconditions: The system always displays the menu of available functions.

Postconditions: The door is closed and lock armed & the auto-lock timer is reset.

Flow of Events for Main Success Scenario:

- 1. **Tenant/Landlord** selects the menu item "Lock"
- System** (a) signals affirmation, e.g., "lock armed," (b) signals to **LockDevice** to arm the lock (if
- ← 2. not already armed), (c) signal to **Timer** to reset the auto-lock counter, and (d) signals to **LightSwitch** to turn the light off (?)

Flow of Events for Extensions (Alternate Scenarios):

2a. System senses that the door is not closed, so the lock cannot be armed

- ← 1. **System** (a) signals a warning that the door is open, and (b) signal to **Timer** to start the alarm counter
- 2. **Tenant/Landlord** closes the door
- System** (a) senses the closure, (b) signals affirmation to the **Tenant/Landlord**, (c) signals to
- ← 3. **LockDevice** to arm the lock, (d) signal to **Timer** to reset the auto-lock counter, and (e) signal to **Timer** to reset the alarm counter

Use Case 3: Add User

Use Case UC-3: AddUser

Related Requirements: REQ6 stated in **Error! Reference source not found.**

Initiating Actor: Landlord

Actor's Goal: To register new or remove departed residents at runtime.

Participating Actors: Tenant

Preconditions: None worth mentioning. (But note that this use case is only available on the main computer and not at the door keypad.)

Postconditions: The modified data is stored into the database.

Flow of Events for Main Success Scenario:

- 1. **Landlord** selects the menu item “ManageUsers”
- 2. **Landlord** identification: Include Login (UC-8)
- ← 3. **System** (a) displays the options of activities available to the Landlord (including “Add User” and “Remove User”), and (b) prompts the **Landlord** to make selection
- 4. **Landlord** selects the activity, such as “Add User,” and enters the new data
- ← 5. **System** (a) stores the new data on a persistent storage, and (b) signals completion

Flow of Events for Extensions (Alternate Scenarios):

- 4a. Selected activity entails adding new users: Include AddUser (UC-3)
- 4b. Selected activity entails removing users: Include RemoveUser (UC-4)

Use Case 5: Inspect Access History

Use Case UC-5: Inspect Access History

Related Requirements: REQ8 and REQ9 stated in Table 2-1

Initiating Actor: Any of: Tenant, Landlord

Actor's Goal: To examine the access history for a particular door.

Participating Actors: Database, Landlord

Preconditions: **Tenant/Landlord** is logged in the system and is shown a hyperlink “View Access History.”

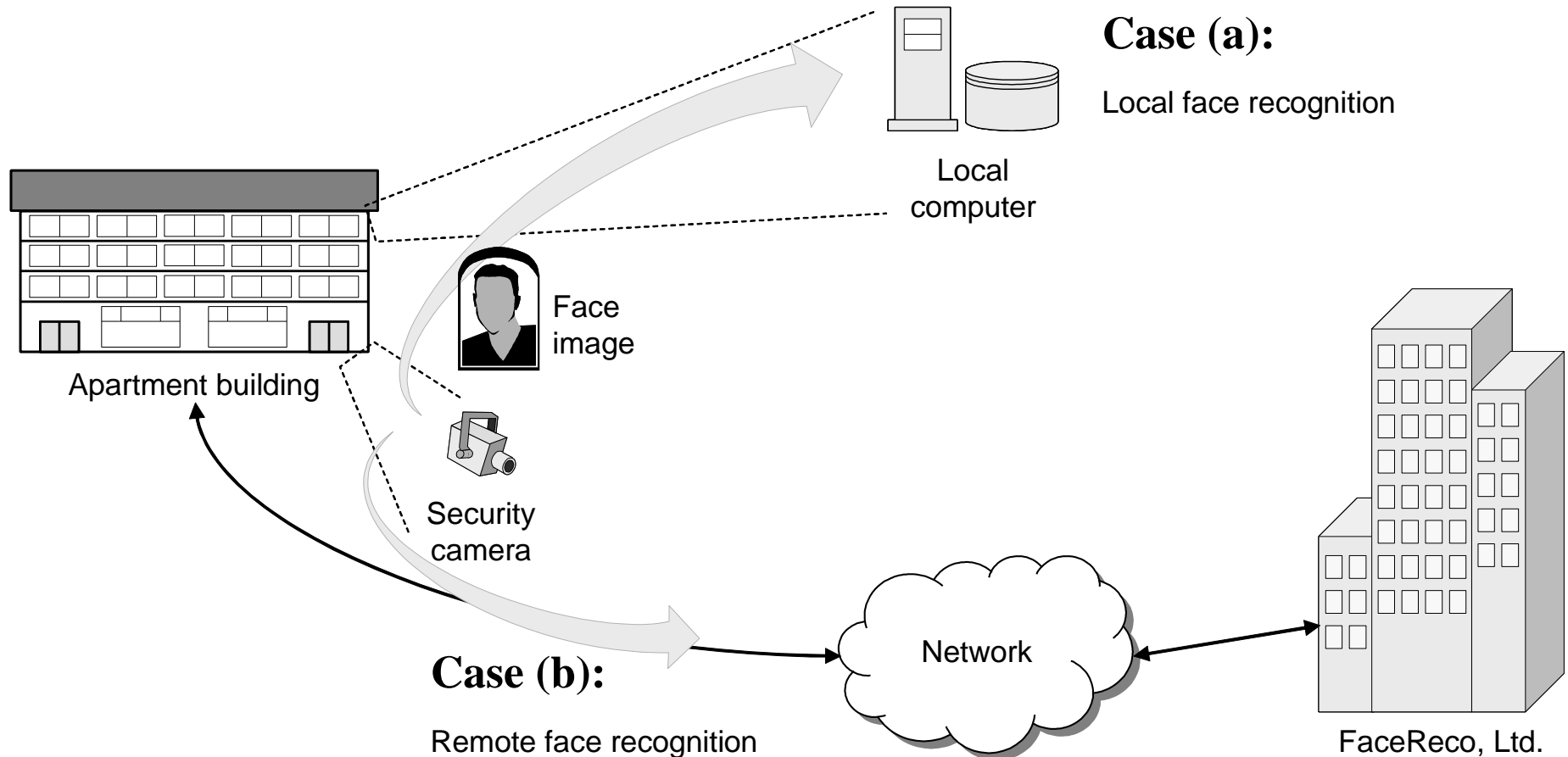
Postconditions: None.

Flow of Events for Main Success Scenario:

- 1. **Tenant/Landlord** clicks the hyperlink “View Access History”
- ← 2. **System** prompts for the search criteria (e.g., time frame, door location, actor role, event type, etc.) or “Show all”
- 3. **Tenant/Landlord** specifies the search criteria and submits
- ← 4. **System** prepares a database query that best matches the actor's search criteria and retrieves the records from the **Database**
- 5. **Database** returns the matching records
- ↷ 6. **System** (a) additionally filters the retrieved records to match the actor's search criteria; (b) renders the remaining records for display; and (c) shows the result for **Tenant/Landlord**'s consideration
- 7. **Tenant/Landlord** browses, selects “interesting” records (if any), and requests further investigation (with an accompanying complaint description)
- ↷ 8. **System** (a) displays only the selected records and confirms the request; (b) archives the request in the **Database** and assigns it a tracking number; (c) notifies **Landlord** about the request; and (d) informs **Tenant/Landlord** about the tracking number
- ←

System Boundary & Subsystems

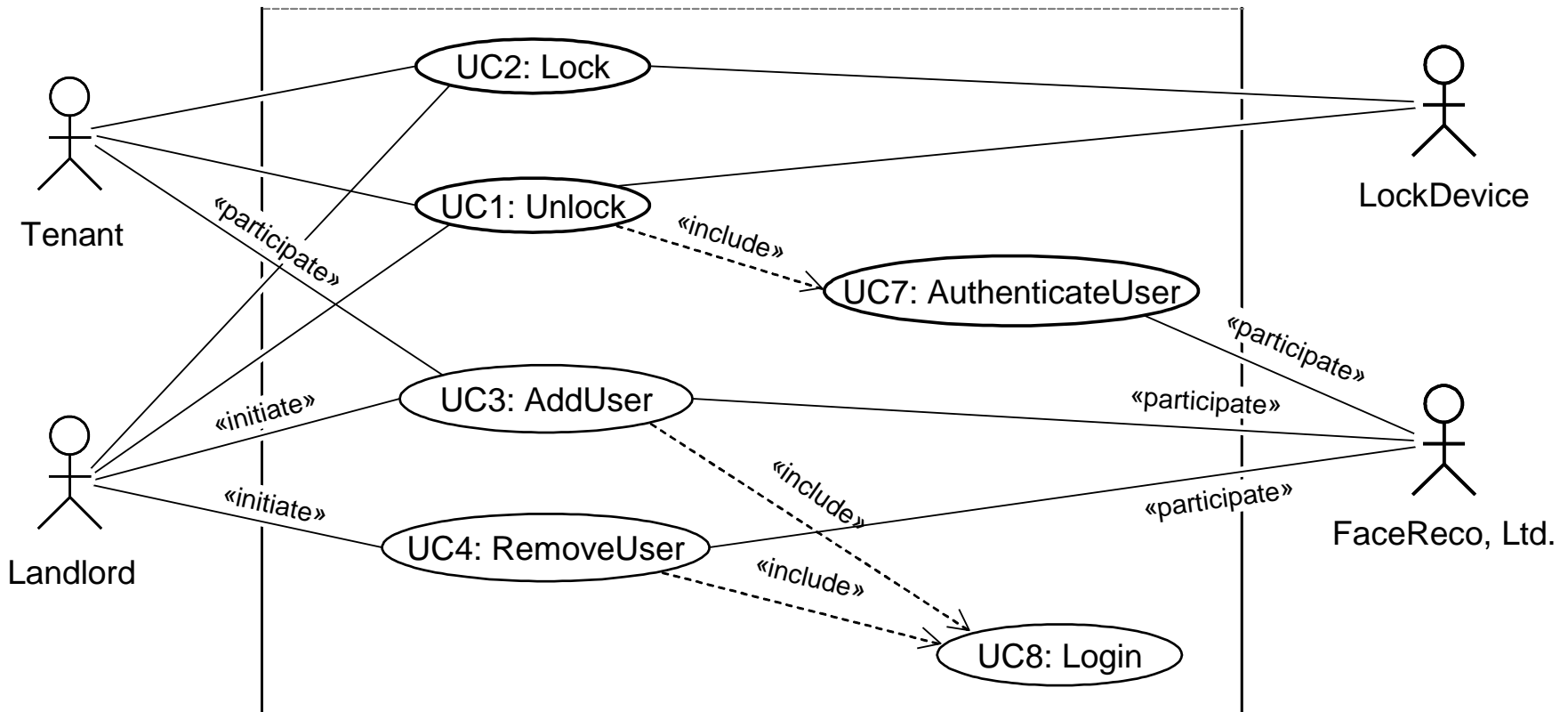
Use Case Variations Example:



Modified Use Case Diagram

Authentication subsystem (FaceReco, Ltd.)
is externalized from the system-to-be:

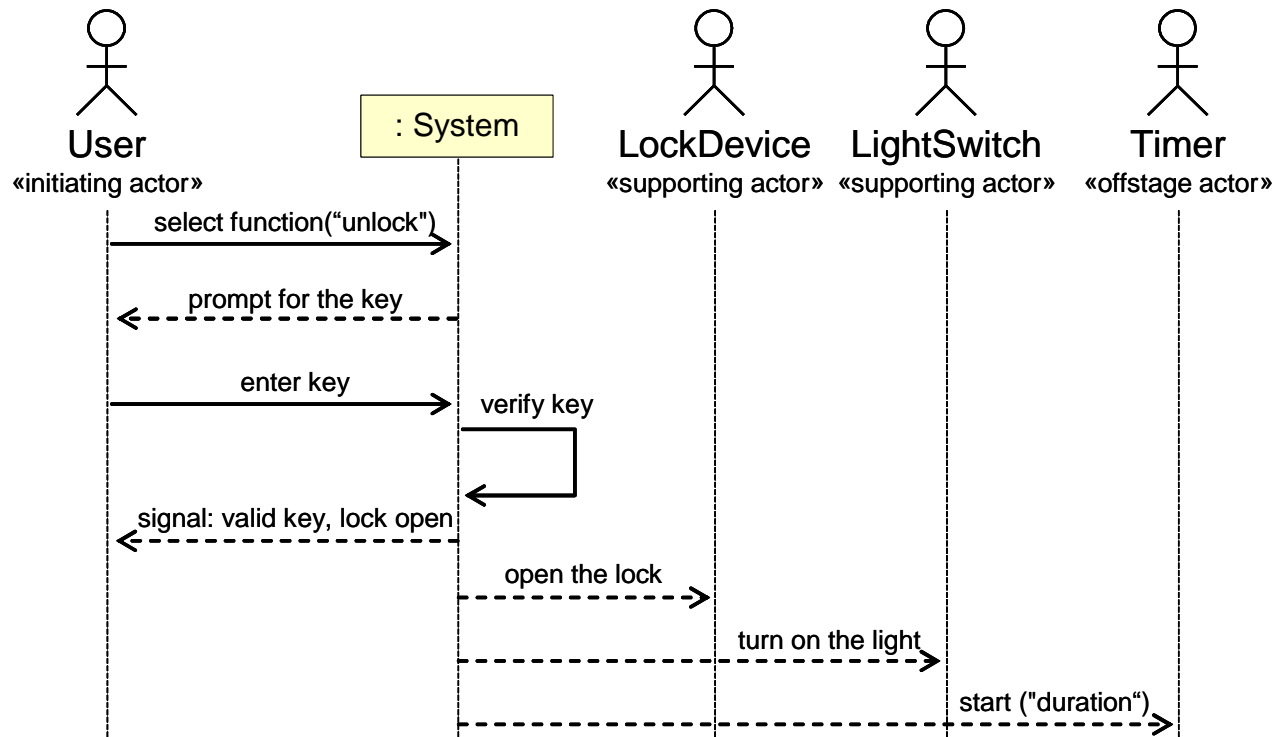
UC1: Unlock
UC2: Lock
UC3: AddUser
UC4: RemoveUser
UC5: InspectAccessHistory
UC6: SetDevicePrefs
UC7: AuthenticateUser
UC8: Login



System Sequence Diagram

[Modeling System Workflows]

Use case: Unlock



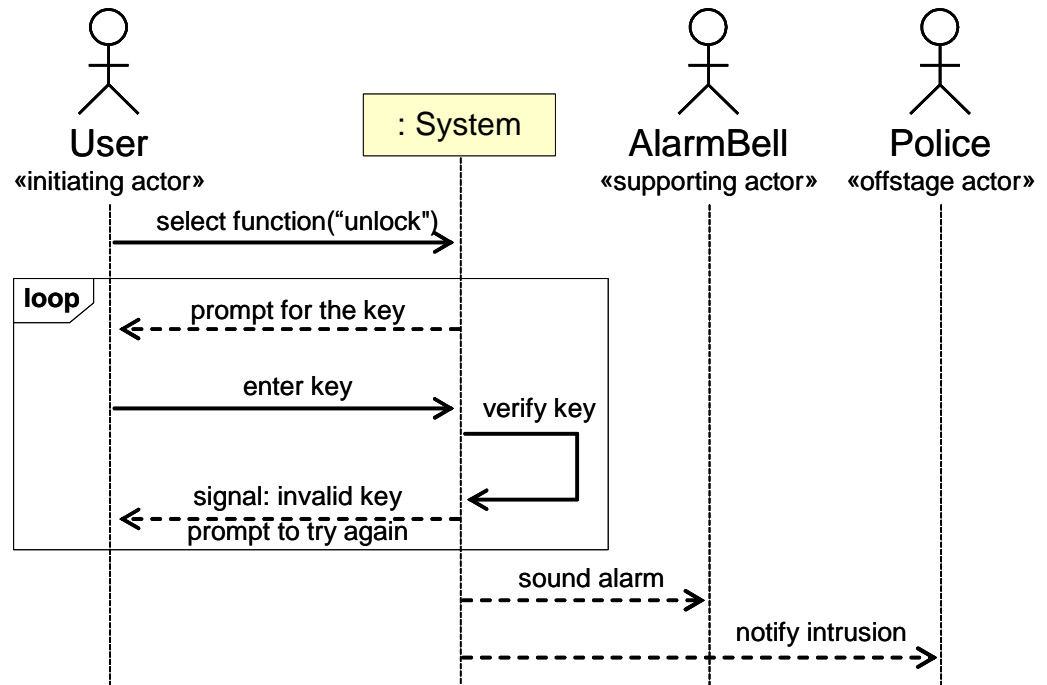
Main success scenario

Similar to UML sequence diagrams,
but for **actor interactions** instead of software **object interactions**

System Sequence Diagram

[Modeling System Workflows]

Use case: Unlock



Alternate scenario (burglary attempt)

Activity Diagram

[Modeling System Workflows]

