Lecture 03: Requirements Engineering – Part 2

Dietmar Pfahl

email: dietmar.pfahl@ut.ee

Fall 2017
ICS Day 2017 – Friday, 29 September

When? – 14:15
Where? – Paabel (Ülikooli 17)
What? –
  To bring together students and academic/research staff
  To introduce interesting research directions that are carried out in the institute
  To network and find topics for Bachelor's and Master's theses

More info:
Schedule of Lectures (Tentative)

Week 01: Introduction to SE
Week 02: Requirements Engineering I
**Week 03: Requirements Engineering II**
Week 04: Analysis
Week 05: Development Infrastructure I
Week 06: Development Infrastructure II
Week 07: Architecture and Design
Week 08: Refactoring
Week 09: Verification and Validation I
Week 10: Crowdsourced Testing
Week 11: Continuous Development and Integration
Week 12: Agile/Lean Methods
Week 13: Software Craftsmanship
Week 14: Course wrap-up, review and exam preparation
Week 15: no lecture
Homework 1 Submission

Make sure you submit your Homework 1 solution on time via your team project wiki on Bitbucket!

(double-check YOUR submission deadline: it’s at midnight before your lab takes place)

Penalties apply for late submission!
No exceptions will be allowed!
Labs Next Week

Lab 2 Assignment:

1. Project Planning
   - Refinement of requirements into Tasks
   - Prioritization of Tasks (P1, P2, P3, P4)
   - Responsibility assignment
   - Effort estimation

2. Five Use Case Descriptions

3. (Initial) Domain Model
Goal of this Lecture: To give answers to the following questions …

- What is ‘Requirements Engineering’?
- Why is RE important?
- Why is RE difficult?
- Who is involved in RE?
- What are ‘Requirements’?
- What types of requirements exist?
- What levels of requirements exist?
- What process steps are involved in RE?
- How to get started with RE?
- How to elicit requirements?
- How to represent/document requirements?
- How to use requirements for project planning?
Acknowledgements

Textbooks/Slides:

Representation Styles

- Natural language (plus supporting tables and graphs)
- Structured natural language / Scenarios
  - e.g., use case descriptions, user stories, CRC cards, ...
- Semi-formal notations
  - e.g., UML diagrams (use case diagrams, class diagrams, state diagram, sequence charts, etc.)
- Formal notations (with formal semantics)
  - e.g., abstract model-based (VDM, Larch, B, ...) or algebraic (OBJ, EJB, ACT-ONE, ...)

Not covered in this course
Example: Home Access Control

Objective: Design an electronic system for:

- Home access control
- Locks and lighting operation
- Intrusion detection and warning

Please read: Ch 1.2 / 1.3.1 / 2.2
## Example NL Requirements

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Priority</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ1</td>
<td>5</td>
<td>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).</td>
</tr>
<tr>
<td>REQ2</td>
<td>2</td>
<td>The system shall lock the door when commanded by pressing a dedicated button.</td>
</tr>
<tr>
<td>REQ3</td>
<td>5</td>
<td>The system shall, given a valid key code, unlock the door and activate other devices.</td>
</tr>
<tr>
<td>REQ4</td>
<td>4</td>
<td>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.</td>
</tr>
<tr>
<td>REQ5</td>
<td>2</td>
<td>The system shall maintain a history log of all attempted accesses for later review.</td>
</tr>
<tr>
<td>REQ6</td>
<td>2</td>
<td>The system should allow adding new authorized persons at runtime or removing existing ones.</td>
</tr>
<tr>
<td>REQ7</td>
<td>2</td>
<td>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.</td>
</tr>
<tr>
<td>REQ8</td>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>REQ9</td>
<td>1</td>
<td>The system should allow filing inquiries about “suspicious” accesses. This function shall be available over the Web.</td>
</tr>
</tbody>
</table>
### Example NL Requirements

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Priority</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ1</td>
<td>5</td>
<td>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).</td>
</tr>
<tr>
<td>REQ2</td>
<td>2</td>
<td>The system shall lock the door when commanded by pressing a dedicated button.</td>
</tr>
<tr>
<td>REQ3</td>
<td>5</td>
<td>The system shall, given a valid key code, unlock the door and activate other devices.</td>
</tr>
<tr>
<td>REQ4</td>
<td>4</td>
<td>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.</td>
</tr>
<tr>
<td>REQ5</td>
<td>2</td>
<td>The system shall maintain a history log of all attempted accesses for later review.</td>
</tr>
<tr>
<td>REQ6</td>
<td>2</td>
<td>The system should allow adding new authorized persons at runtime or removing existing ones.</td>
</tr>
<tr>
<td>REQ7</td>
<td>2</td>
<td>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.</td>
</tr>
<tr>
<td>REQ8</td>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>REQ9</td>
<td>1</td>
<td>The system should allow filing inquiries about “suspicious” accesses. This function shall be available over the Web.</td>
</tr>
</tbody>
</table>

* ‘shall’: mandatory (?)

* ‘should’: optional (?)
Example NL Requirements

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Priority</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ1</td>
<td>5</td>
<td>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).</td>
</tr>
<tr>
<td>REQ2</td>
<td>2</td>
<td>The system shall lock the door when commanded by pressing a dedicated button.</td>
</tr>
<tr>
<td>REQ3</td>
<td>5</td>
<td>The system shall, given a valid key code, unlock the door and activate other devices.</td>
</tr>
<tr>
<td>REQ4</td>
<td>4</td>
<td>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.</td>
</tr>
<tr>
<td>REQ5</td>
<td>2</td>
<td>The system shall maintain a history log of all attempted accesses for later review.</td>
</tr>
<tr>
<td>REQ6</td>
<td>2</td>
<td>The system should allow adding new authorized persons at runtime or removing existing ones.</td>
</tr>
<tr>
<td>REQ7</td>
<td>2</td>
<td>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.</td>
</tr>
<tr>
<td>REQ8</td>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>REQ9</td>
<td>1</td>
<td>The system should allow filing inquiries about “suspicious” accesses. This function shall be available over the Web.</td>
</tr>
</tbody>
</table>

'Compound' REQ: How test it?
**Example NL Requirements**

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Priority</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ1</td>
<td>5</td>
<td>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).</td>
</tr>
<tr>
<td>REQ2</td>
<td>2</td>
<td>The system shall lock the door when commanded by pressing a dedicated button.</td>
</tr>
<tr>
<td>REQ3</td>
<td>5</td>
<td>The system shall, given a valid key code, unlock the door and activate other devices.</td>
</tr>
<tr>
<td>REQ4</td>
<td>4</td>
<td>For REQ3, the customer may suggest these test cases:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Test with the valid key of a current tenant on his or her apartment (pass)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Test with the valid key of a current tenant on someone else’s apartment (fail)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Test with an invalid key on any apartment (fail)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Test with the key of a removed tenant on his or her previous apartment (fail)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Test with the valid key of a just-added tenant on his or her apartment (pass)</td>
</tr>
<tr>
<td>REQ5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>REQ6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>REQ7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>REQ8</td>
<td>1</td>
<td>The system should allow filing inquiries about “suspicious” accesses. This function shall be available over the Web.</td>
</tr>
<tr>
<td>REQ9</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

*LTAT.05.003 / Lecture 03 / © Dietmar Pfahl 2017*
User Stories

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

- user-role (benefactor)
- capability (functionality)
- business-value (motivation/rationale)

**who** - **what** - **why**

- Similar to NL requirements, but focus on the user benefits, instead on system characteristics (alone).
- Unfortunately, third element (business-value) is often omitted
- Preferred tool in agile methods.
NL Requirements vs. User Stories

Traditional requirement – “shall” statements:

• “The system shall provide a user configurable interface for all user and system manager functions”

• “The user interface shall be configurable in the areas of:
  • Screen layout
  • Font
  • Background and text color

Corresponding “User Story”:

• “As a system user or system manager, …

• … I want to be able to configure the user interface for screen layout, font, background color, and text color, …

• … so that I can use the system in the most efficient manner”

who - what - why
Example User Stories

<table>
<thead>
<tr>
<th>Identifier</th>
<th>User Story</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST-1</td>
<td>As an authorized person (tenant or landlord), I can keep the doors locked at all times. (... so I am safe)</td>
</tr>
<tr>
<td>ST-2</td>
<td>As an authorized person (tenant or landlord), I want the lock be automatically locked after a defined period of time.</td>
</tr>
<tr>
<td>ST-3</td>
<td>As an authorized person (tenant or landlord), I can lock the doors on demand.</td>
</tr>
<tr>
<td>ST-4</td>
<td>As an authorized person (tenant or landlord), I can unlock the doors. <em>(Test: Allow a small number of mistakes, say three.)</em></td>
</tr>
<tr>
<td>ST-5</td>
<td>As a landlord, I can at runtime manage authorized persons.</td>
</tr>
<tr>
<td>ST-6</td>
<td>As an authorized person (tenant or landlord), I can view past accesses.</td>
</tr>
<tr>
<td>ST-7</td>
<td>As a tenant, I can configure the preferences for activation of various devices.</td>
</tr>
<tr>
<td>ST-8</td>
<td>As a tenant, I can file complaint about “suspicious” accesses.</td>
</tr>
</tbody>
</table>

Note: ‘Why’ part is missing in the examples above.
Time Estimation with User Story Points

- Points assigned to individual user stories
- Total work size estimate:

\[
\text{Total size} = \sum \text{points-for-story}_i \quad (i = 1..N)
\]

- Velocity (= Productivity) estimated from experience
- Estimate the work duration:

\[
\text{Project duration} = \frac{\text{Total size}}{\text{Velocity}} \quad \text{[time unit]}
\]
# Example User Stories

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

<table>
<thead>
<tr>
<th>Identifier</th>
<th>User Story</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST-1</td>
<td>...</td>
<td>4 points</td>
</tr>
<tr>
<td>ST-2</td>
<td>...</td>
<td>3 pts</td>
</tr>
<tr>
<td>ST-3</td>
<td>...</td>
<td>6 pts</td>
</tr>
<tr>
<td>ST-4</td>
<td>...</td>
<td>9 pts</td>
</tr>
<tr>
<td>ST-5</td>
<td>...</td>
<td>10 pts</td>
</tr>
<tr>
<td>ST-6</td>
<td>...</td>
<td>6 pts</td>
</tr>
<tr>
<td>ST-7</td>
<td>...</td>
<td>6 pts</td>
</tr>
<tr>
<td>ST-8</td>
<td>...</td>
<td>6 pts</td>
</tr>
</tbody>
</table>

Sum: 50 User Story Points (USP)

Team: 3 persons

How long will it take?
Example Estimation

- Points assigned to individual user stories
- Total work size estimate:
  \[ \text{Total size} = \sum \text{points-for-story } i \quad (i = 1..N) \]
- Velocity (= Productivity) estimated from experience
- Estimate the work duration:
  \[ \text{Project duration} = \frac{\text{Total size}}{\text{Velocity}} \quad [\text{time unit}] \]

- 8 User Stories
- 50 US Points
- Average Team Size: 6 persons
- Average Project Size: 600 USPs
- Average Project Duration: 30 days
- Average Velocity (6 pers): 20 USP/day
  \[ = 3.33 \text{ USP/person-day} \]
- 50 US Points & 3 Pers. Team:
  Duration: ? days
Example Estimation

- Points assigned to individual user stories
- Total work size estimate:
  \[ \text{Total size} = \sum \text{points-for-story}_i \quad (i = 1..N) \]
- Velocity (= Productivity) estimated from experience
- Estimate the work duration:
  \[ \text{Project duration} = \frac{\text{Total size}}{\text{Velocity}} \quad \text{[time unit]} \]

- 8 User Stories
- 50 US Points
- Average Team Size: 6 persons
- Average Project Size: 600 USPs
- Average Project Duration: 30 days
- Average Velocity (6 pers): 20 USP/day = 3.33 USP/person-day
- 50 US Points & 3 Pers. Team: Duration: 50/(3*3.33) = 5 days
Agile Project Estimation/Planning

**Work backlog**

1) ST-4: Unlock 2.7 day (9pts)
2) ST-2: Auto-Lock 0.9 day (3pts)
3) ST-1: Keep locked 1.2 day (4pts)
4) ST-5: Manage Users 3 day (10pts)
5) ST-6: View History 1.8 day (6pts)

- List prioritized by the customer
- Estimated work duration
- Assumes that 1 person works fulltime on this task without interruption
- Items pulled by the team into an iteration

- 20 days
- 1st iteration
- 2nd iteration
- n-th iteration
- Estimated completion date
- Time

List prioritized by the customer

5 days (40 hours)
User Stories versus Tasks

How to split User Stories into Tasks:
https://www.youtube.com/watch?v=gZ4uLafsxAk

User Story = Point of view of system user (What?)
Task = Point of view of system developer (How?)
Use Cases

• For Functional Requirements Analysis and Specification

• A **use case** is a description of how a user will use the system-to-be to achieve business goals

  • Detailed use cases are usually written as **usage scenarios** or **scripts**, listing a specific sequence of actions and interactions between the actors and the system
Scenarios

- Scenario = real-life example of how a system can be used
- They should include
  - A description of the starting situation (state)
  - A description of the normal flow of events
  - A description of what can go wrong
  - Information about other concurrent activities
  - A description of the state when the scenario finishes
Use Case Diagrams and Descriptions

Use Case Description:
- Name of Use Case
- Actors associated with Use Case
- Pre-conditions
- Post-conditions
- Normal Flow of Events (Basic Scenario)
- Alternative Flow of Events (Alternative Scenarios)
- ...

Diagram: Use Case Model
- Actors
- Use Cases
- Use-Case Descriptions

...
Types of Actors

• **Initiating actor** (also called *primary actor* or “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:
  • **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)
Identifying Actors

- **Ask the following questions:**
  - Who will be a primary user of the system? (primary actor)
  - Who will need support from the system to do her daily tasks?
  - Who will maintain, administrate, keep the system working? (secondary actor)
  - Which hardware devices does the system need?
  - With which other systems does the system need to interact with?
  - Who or what has an interest in the results that the system produces?

- **Look for:**
  - the users who directly use the system
  - also others who need services from the system
Finding Use Cases

- For each actor, ask the following questions:
  - Which functions does the actor require from the system?
  - What does the actor need to do?
  - Does the actor need to read, create, destroy, modify, or store some kinds of information in the system?
  - Does the actor have to be notified about events in the system?
  - Does the actor need to notify the system about something?
  - What do those events require in terms of system functionality?
  - Could the actor’s daily work be simplified or made more efficient through new functions provided by the system?
# Deriving Use Cases from System Requirements

### Use Cases Table

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

(Actors are often given, if working from user stories instead of ‘shall/should’-statements.)
# Traceability Matrix

**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

<table>
<thead>
<tr>
<th>Req’t</th>
<th>PW</th>
<th>UC1</th>
<th>UC2</th>
<th>UC3</th>
<th>UC4</th>
<th>UC5</th>
<th>UC6</th>
<th>UC7</th>
<th>UC8</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ1</td>
<td>5</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>REQ3</td>
<td>5</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ5</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>REQ6</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>REQ7</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ8</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ9</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Purpose:**
- Check all REQs are covered by UCs
- Check no UC is added that doesn’t have a REQ
- Prioritize UCs

**PW = Priority Weight**

<table>
<thead>
<tr>
<th>Max PW</th>
<th>5</th>
<th>5</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total PW</td>
<td>16</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>
Use Case Diagram: Device Control

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

First tier use cases
- UC1: Unlock
- UC2: Lock

Second tier use cases
- UC7: AuthenticateUser

Actors:
- Tenant
- Landlord

System boundary

Communication

Use case
Use Case Diagrams and Descriptions

Use Case Description:
Name of Use Case
Actors associated with Use Case
Pre-conditions
Post-conditions
Normal Flow of Events (Basic Scenario)
Alternative Flow of Events (Alternative Scenarios)

...
### Schema for Use Case Description

<table>
<thead>
<tr>
<th><strong>Use Case UC-#</strong></th>
<th><strong>Name / Identifier</strong></th>
<th><strong>[verb phrase]</strong></th>
</tr>
</thead>
</table>

**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

<table>
<thead>
<tr>
<th>Use Case UC-1: Unlock</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Related Requirem’ts:</strong></td>
</tr>
<tr>
<td><strong>Initiating Actor:</strong></td>
</tr>
<tr>
<td><strong>Actor’s Goal:</strong></td>
</tr>
<tr>
<td><strong>Participating Actors:</strong></td>
</tr>
</tbody>
</table>

- 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» UC-7

<table>
<thead>
<tr>
<th>Use Case UC-7:</th>
<th>AuthenticateUser (sub-use case)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related Requirements:</td>
<td>REQ3, REQ4</td>
</tr>
<tr>
<td>Initiating Actor:</td>
<td>Any of: Tenant, Landlord</td>
</tr>
<tr>
<td>Actor’s Goal:</td>
<td>To be positively identified by the system (at the door interface).</td>
</tr>
<tr>
<td>Participating Actors:</td>
<td>AlarmBell, Police</td>
</tr>
<tr>
<td>Preconditions:</td>
<td>• The set of valid keys stored in the system database is non-empty.</td>
</tr>
<tr>
<td></td>
<td>• The counter of authentication attempts equals zero.</td>
</tr>
<tr>
<td>Postconditions:</td>
<td>None worth mentioning.</td>
</tr>
</tbody>
</table>

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 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
Use Case Diagram: Account Management

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

Account Management Subsystem

Tenant

Landlord
Use Case Diagram: Account Management

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

Why secondary?
‘Login’ Use Case?

BAD:

```
Landlord
  +----------------+
  | Login          |
  +----------------+
  | AddUser        |
  +----------------+
  | SetDevicePrefs |
```

GOOD:

```
Landlord
  +----------------+
  | AddUser        |
  +----------------+
  | SetDevicePrefs |
  +----------------+
  +----------------+
  | Login          |
      «include»
```

Novice developers frequently identify user login as a use case.
Expert developers argue that login is not a use case in its own right.

Recall that use case is motivated by user’s goal;
The user initiates interaction with the system to achieve a certain goal. You
are not logging in for the sake of logging in—you are logging in to do some
work, and this work is your use case.
Optional Use Cases: «extend»

Example optional use cases:

Key differences between «include» and «extend» relationships

<table>
<thead>
<tr>
<th></th>
<th>Included use case</th>
<th>Extending use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is this use case optional?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Is the base use case complete</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>without this use case?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is the execution of this use</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>case conditional?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does this use case change the</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>behavior of the base use case?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where do we want to take you? (by the end of this course)

Sample student software projects (Fall 2015)

- https://github.com/vladislavivanistsev/SuperCap/
- https://github.com/alajal/license-management/
Project Estimation

with

Use Cases
Use Case Points

For the sum of all Use Cases:

- UCP equation is composed of four variables:
  - Unadjusted Use Case Point (UUCP)
  - The Technical Complexity Factor (TCF)
  - The Environment Complexity Factor (ECF)

(adjusted) \[ UCP = UUCP \times TCF \times ECF \]

- UUCP is the sum of Unadjusted Actor Weight (UAW) and Unadjusted Use Case Weight (UUCW).

\[ UUCP = UAW + UUCW \]
# Unadjusted Actor Weight

<table>
<thead>
<tr>
<th>Actor Type</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Communicates to system through API</td>
<td>1</td>
</tr>
<tr>
<td>Average</td>
<td>Interacts with the system through some protocol (HTTP, FTP, or probably some user defined protocol), or Are data stores (Files, DBMS)</td>
<td>2</td>
</tr>
<tr>
<td>Complex</td>
<td>Interacts through HCI (GUI)</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
\text{UAW} = (\text{Total No. of Simple actors } \times 1) + (\text{Total No. Average actors } \times 2) + (\text{Total No. Complex actors } \times 3)
\]
## Unadjusted Use Case Weight

<table>
<thead>
<tr>
<th>Use Case Type</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>1 to 3 transactions</td>
<td>5</td>
</tr>
<tr>
<td>Average</td>
<td>4 to 7 transactions</td>
<td>10</td>
</tr>
<tr>
<td>Complex</td>
<td>8 or more transactions</td>
<td>15</td>
</tr>
</tbody>
</table>

\[
UUCW = (\text{Total No. of Simple Use Cases} \times 5) + (\text{Total No. Average Use Case} \times 10) + (\text{Total No. Complex Use Cases} \times 15)
\]
Technical Complexity Factor – TCF

<table>
<thead>
<tr>
<th>Technical Factor</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF(1)</td>
<td>Distributed System</td>
<td>2</td>
</tr>
<tr>
<td>TF(2)</td>
<td>Performance</td>
<td>1</td>
</tr>
<tr>
<td>TF(3)</td>
<td>End User Efficiency</td>
<td>1</td>
</tr>
<tr>
<td>TF(4)</td>
<td>Complex Internal Processing</td>
<td>1</td>
</tr>
<tr>
<td>TF(5)</td>
<td>Reusability</td>
<td>1</td>
</tr>
<tr>
<td>TF(6)</td>
<td>Installability</td>
<td>0.5</td>
</tr>
<tr>
<td>TF(7)</td>
<td>Usability</td>
<td>0.5</td>
</tr>
<tr>
<td>TF(8)</td>
<td>Portability</td>
<td>2</td>
</tr>
<tr>
<td>TF(9)</td>
<td>Modifiability</td>
<td>1</td>
</tr>
<tr>
<td>TF(10)</td>
<td>Concurrency</td>
<td>1</td>
</tr>
<tr>
<td>TF(11)</td>
<td>Includes special security requirements</td>
<td>1</td>
</tr>
<tr>
<td>TF(12)</td>
<td>Provides direct access by third parties</td>
<td>1</td>
</tr>
<tr>
<td>TF(13)</td>
<td>Special User training facilities are required</td>
<td>1</td>
</tr>
</tbody>
</table>

Each TF(i) can have a value from 0 (factor is irrelevant) to 5 (factor is essential)

TCF = 0.6 + TF/100 with \( \sum_{i=1}^{13} (TF(i) \times Weight(i)) \)
## Environmental Complexity Factor - ECF

<table>
<thead>
<tr>
<th>Factor Number</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF(1)</td>
<td>Familiarity with system development process in use</td>
<td>1.5</td>
</tr>
<tr>
<td>EF(2)</td>
<td>Application experience</td>
<td>0.5</td>
</tr>
<tr>
<td>EF(3)</td>
<td>Object-oriented experience</td>
<td>1.0</td>
</tr>
<tr>
<td>EF(4)</td>
<td>Lead analyst capability</td>
<td>0.5</td>
</tr>
<tr>
<td>EF(5)</td>
<td>Motivation</td>
<td>1.0</td>
</tr>
<tr>
<td>EF(6)</td>
<td>Requirements stability</td>
<td>2.0</td>
</tr>
<tr>
<td>EF(7)</td>
<td>Part time staff</td>
<td>-1.0</td>
</tr>
<tr>
<td>EF(8)</td>
<td>Difficulty of programming language</td>
<td>-1.0</td>
</tr>
</tbody>
</table>

Each EF(i) can have a value from 0 (no experience) to 5 (expert)

Complexity increases, the smaller EF(1) to EF(6) and the greater EF(7) & EF(8)

ECF = 1.4 + (-0.03*EF) with EF = \( \sum_{i=1}^{8} (EF(i) \times Weight(i)) \)
Use Case Points

For the sum of all Use Cases:

- UCP equation is composed of four variables:
  - Unadjusted Use Case Point (UUCP)
  - The Technical Complexity Factor (TCF)
  - The Environment Complexity Factor (ECF)

\[
UCP = UUCP \times TCF \times ECF
\]

- UUCP is the sum of Unadjusted Actor Weight (UAW) and Unadjusted Use Case Weight (UUCW).

\[
UUCP = UAW + UUCW
\]
Use Case Points

For the sum of all Use Cases:

- UCP equation is composed of four variables:
  - Unadjusted Use Case Point (UUCP)
  - The Technical Complexity Factor (TCF)
  - The Environment Complexity Factor (ECF)
  - The Productivity Factor (PF)

Effort (UCP) = UUCP * TCF * ECF * PF

- UUCP is the sum of Unadjusted Actor Weight (UAW) and Unadjusted Use Case Weight (UUCW).

UUCP = UAW + UUCW
Use Case Points: Project Effort

For the sum of all Use Cases:

- UCP equation is composed of four variables:
  - Unadjusted Use Case Point (UUCP)
  - The Technical Complexity Factor (TCF)
  - The Environment Complexity Factor (ECF)
  - The Productivity Factor (PF)

\[
\text{Eff(UCP)} = \text{UUCP} \times \text{TCF} \times \text{ECF} \times \text{PF} \quad [\text{person-hours}]
\]

- UUCP is the sum of Unadjusted Actor Weight (UAW) and Unadjusted Use Case Weight (UUCW).

\[
\text{UUCP} = \text{UAW} + \text{UUCW}
\]

Either 20 person-hours/UCP or 28 person-hours/UCP
Productivity Factor - PF

If sum of [(number of factors E1 through E6 assigned value < 3) and (number of factors E7 and E8 assigned value > 3)] ≤ 2
PF = 20 ph/UCP

Else
If sum of [(number of factors E1 through E6 assigned value < 3) and (number of factors E7 and E8 assigned value > 3)] = 3 or 4
PF = 28 ph/UCP

Else: Rethink project; it has too high a risk of failure

Example: http://en.wikipedia.org/wiki/Use_Case_Points
Next Lecture

• Date/Time:
  • Friday, 29 Sep, 10:15-12:00

• Topic:
  • Analysis

• For you to do:
  • Make sure you submit your Lab Task1 solution on time (double-check submission deadlines!)
  • Go to next lab → Second HW Assignment