Requirements Specifications, Use Case Diagrams

PB007 Software Engineering I

Bruno Rossi

27. 09. 2017

Functional Requirements

Functional requirements are the expression of what the system should **do** related to the functionality of the system.
It is recommended to write it in the form:

- <**id**><**system**><**function**>

Examples:

- 1. The ATM verifies the validity of the inserted card.
- 2. The ATM verifies the PIN of the specified customer.
- 3. The ATM does not allow more than CZK 10,000 on one card within 24 hours.

Non-Functional Requirements

Non-Functional Requirements are in general constraints imposed on the system that are not strictly related to the functionality. We call these also quality requirements, generally we refer to performance, capacity, availability, security constraints.

Examples:

- 1. ATM will be programmed in C++ language.
- 2. ATM will be in communication with the bank with 256-bit encryption.
- 3. ATM needs to take less than 3 seconds to verify the validity of one card.

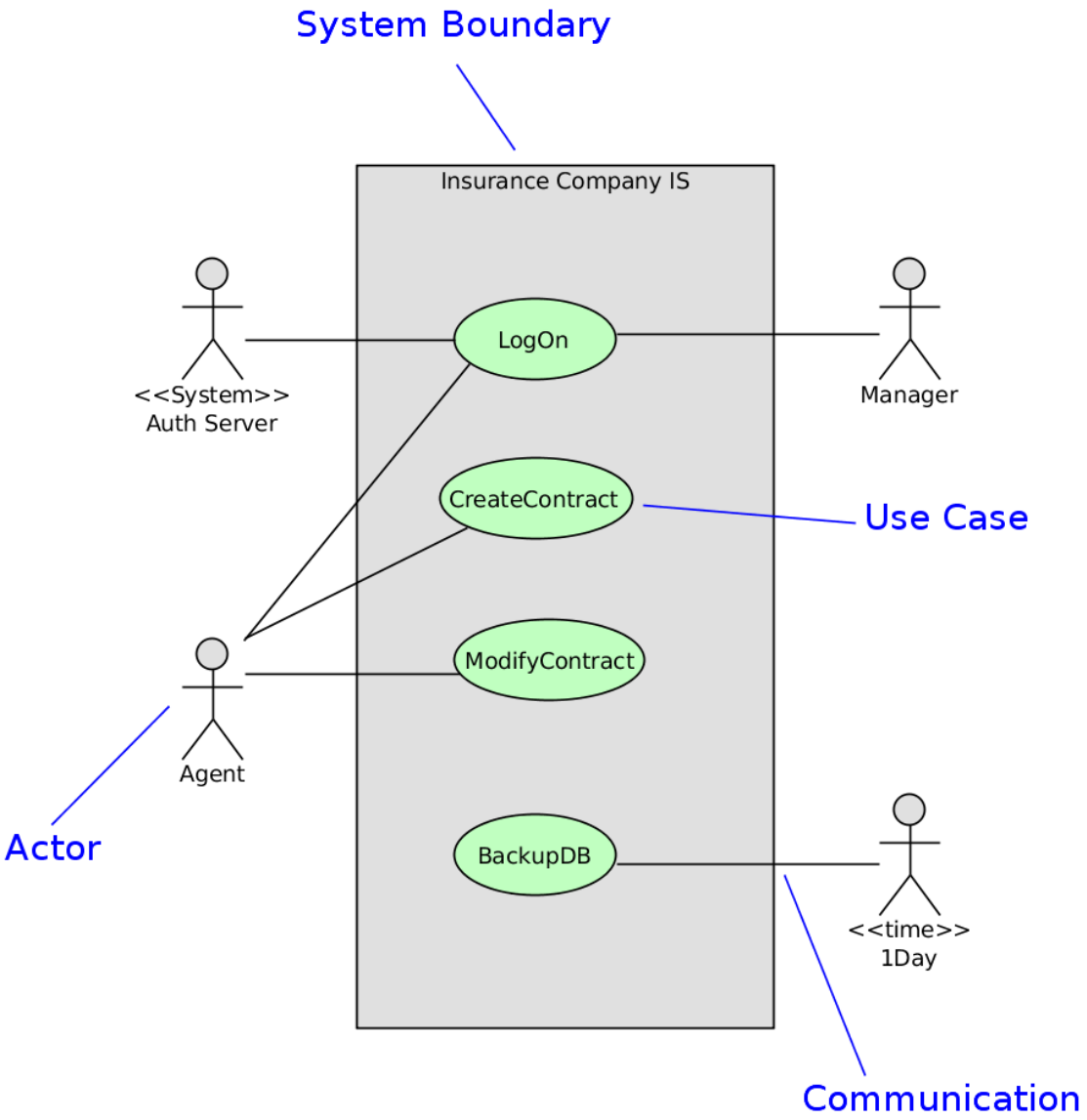
Use Case Diagram

The **Use Case Diagram** is a method of graphical representation of functional requirements of a software system.

It generally shows:

- **Boundaries of the system and subjects**;
- **Actors**;
- **Use Cases**;
- **Relationships among entities**;

Use Case Diagram - Example



Actors I

An **Actor** is a role represented by an external entity to the system (user, another system, devices, time) that directly communicates with the system itself.

- Actors are always external entities to the system (*this is why it is important to define the boundaries of the system*).
- Actors interact directly with the system.
- Actors represent roles played in the system. As such, the same person can be represented by different roles (e.g. admin, standard user).
- One person can have more roles in the system and also such roles can change over time.
- Each actor must have an appropriate naming.
- Every actor shoul have a short description.

Actors II

Identification of actors:

- Who or what uses the system?
- Are there roles involved in the interaction with the system?
- What other systems work together with the system to reach the final goals?
- Who/what obtains/provides information from/to the system?
- Are there some events that occur periodically or at fixed periods of time?

Use Cases I

Each **Use Case** describes the behavior of the system in the interaction with external actors. These are all the activities that actors need to perform interacting with the system.

- The use cases always begin with some action from one actor (*a primary actor*). The interactions can subsequently involve also other actors (*secondary actors*).
- Use Cases are always seen from the point of view of the actors.
- Naming should conform to the conventions in the domain, to avoid misunderstandings.

Use Cases II

Identification of the Use Cases:

- What features are required by a particular actor of the system?
- Is there the need to store and/or retrieve some information? If yes, who will trigger such behaviour?
- What happens during different system status changes? How are actors informed?
- Are there external events that affect the system? what alerts the system for these events?

Suggestion about the Steps to Follow

When you create a Use Case diagram, the following procedure is recommended:

1. Define the system's boundaries;
2. Find the actors;
3. Find the use cases;
4. Determine the relationships among use cases and actors;
5. Specification in more detail of the use cases identified;

Useful Resources

- www.agilemodeling.com/artifacts/useCaseDiagram.htm
- http://www.agilemodeling.com/style/useCaseDiagram.htm
- www.andrew.cmu.edu/course/90-754/umlucdfaq.html
- http://uml.czweb.org/pripad_uziti.htm
- www.drdobbs.com/top-ten-use-case-mistakes/184414701

Tasks

- Create a new project in Visual Paradigm.
  - Format of the project's name: lastname1-lastname2-lastname3.
  - Language: UML 2.x.
- Create a numbered list of functional requirements (possibly sorted by actors) as the main text in the documentation of the Use Case diagram.
- Define several non-functional requirements, especially platform (e.g. Java) reliability requirements (like backup needs), performance-critical parts of the system (expected number of requests per minute), and so on.
- Based on the functional requirements, give an initial Use Case model, i.e. add the actors, the use cases and the relationships.
- Look for gaps in project specifications and ask questions on the information that you are missing.
- Generate a **pdf** file and save the report to the homework vault folder for (**Week 02**).
  - **Deadline:** Monday 02.10.17 23:59

VP Reports Customization