Topic Area: Requirements Engineering

- Introduction
- Requirements Specification
- Desired Properties
- Kinds of Requirements
- Analysis Techniques
- Documents
- Dictionary, Specification
- Specification Languages
- Natural Language
- Decision Tables
- Syntax, Semantics
- Completeness, Consistency, etc.
- Scenarios
- User Stories, Use Cases
- Live Sequence Charts
- Syntax, Semantics
- Working Definition: Software
- Discussion

Example: Vending Machine

Positive scenario: Buy a Softdrink
(i) Insert one 1 euro coin.
(ii) Press the 'softdrink' button.
(iii) Get a softdrink.

Positive scenario: Get Change
(i) Insert one 50 cent and one 1 euro coin.
(ii) Press the 'softdrink' button.
(iii) Get a softdrink.
(iv) Get 50 cent change.

Negative scenario: A Drink for Free
(i) Insert one 1 euro coin.
(ii) Press the 'softdrink' button.
(iii) Do not insert any more money.
(iv) Get two softdrinks.
Notations for Scenarios

The idea of scenarios (sometimes without negative or anti-scenarios) (re-)occurs in many process models or software development approaches.

In the following, we will discuss two-and-a-half notations (in increasing formality):

- **User Stories** (part of Extreme Programming)
- **Use Cases** and Use Case Diagrams (**OOSE**)
- **Sequence Diagrams** (here: Live Sequence Charts (**Damm and Harel, 2001**))

**User Stories** (**Beck, 1999**)

"A User Story is a concise, written description of a piece of functionality that will be valuable to a user (or owner) of the software."

Per user story, use one file card with the user story, e.g. following the pattern:

As a [role] I want [something] so that [benefit].

and in addition:

- unique identifier (e.g. unique number),
- priority (from 1 (highest) to 10 (lowest)) assigned by customer,
- effort, estimated by developers,
- back side of file card: (acceptance) test case(s), i.e., how to tell whether the user story has been realized.

Proposed card layout (front side):

priority, unique identifier, name

estimation, risk

As a [role] I want [something] so that [benefit].

Natural Language Patterns

Natural language requirements can be (tried to be) written as an instance of the pattern "⟨A⟩ ⟨B⟩ ⟨C⟩ ⟨D⟩ ⟨E⟩ ⟨F⟩." (German grammar) where

- A clarifies when and under what conditions the activity takes place
- B is MUST (obligation), SHOULD (wish), or WILL (intention); also: MUST NOT (forbidden)
- C is either "the system" or the concrete name of a (sub-)system
- D one of three possibilities:
  - "does," description of a system activity,
  - "offers," description of a function offered by the system to somebody,
  - "is able if," usage of a function offered by a third party, under certain conditions
- E extensions, in particular an object
- F the actual process word (what happens)

**Example:**

After office hours (=A), the system (=C) should (=B) offer to the operator (=D) a backup (=F) of all new registrations to an external medium (=E).

User Stories: Discussion

- ✔ easy to create, small units
- ✔ close contact to customer
- ✔ objective / testable: by fixing test cases early
- ❌ may get difficult to keep overview over whole system to be developed
  → maybe best suited for changes / extensions (after first iteration).
- ❌ not designed to cover non-functional requirements and restrictions
- ❌ agile spirit: strong dependency on competent developers
- ❌ estimation of effort may be difficult (**Balzert, 2009**)
Use Case: Definition

A use case is a sequence of interactions between an actor (or actors) and a system triggered by a specific actor, which produces a result for an actor. (Jacobson, 1992)

More precisely:

- A use case has participants:
  - the system
  - at least one actor.

- Actor: an actor represents what interacts with the system.

- An actor is a role, which a user or an external system may assume when interacting with the system under design.

- Actors are not part of the system, thus they are not described in detail.

- Actions of actors are non-deterministic (possibly constrained by domain model).

- A use case is triggered by a stimulus as input by the main actor.

- A use case is goal-oriented, i.e., the main actor wants to reach a particular goal.

- A use case describes all interactions between the system and the participating actors that are needed to achieve the goal (or fail to achieve the goal for reasons).

- A use case ends when the desired goal is achieved, or when it is clear that the desired goal cannot be achieved.

Use Case Example

http://commons.wikimedia.org (CC-by-sa 4.0, Dirk Ingo Franke)

name: Authentication

goal: the client wants access to the ATM

pre-condition: the ATM is operational, the welcome screen is displayed, card and PIN of client are available

post-condition: client accepted, services of ATM are offered

post-cond. in exception case: access denied, card returned or withheld, welcome screen displayed

actors: client (main actor), bank system

open questions: none

normal case:

1. client inserts card
2. ATM reads card, sends data to bank system
3. bank system checks validity
4. ATM shows PIN screen
5. client enters PIN
6. ATM reads PIN, sends to bank system
7. bank system checks PIN
8. ATM accepts and shows main menu

exception case 2a: card not readable
   2a.1 ATM displays "card not readable"
   2a.2 ATM returns card
   2a.3 ATM shows welcome screen

exception case 2b: card readable, but not ATM card

exception case 2c: no connection to bank system

exception case 3a: card not valid or disabled

exception case 5a: client cancels

exception case 5b: client doesn't react within 5 s

exception case 6a: no connection to bank system

exception case 7a: first or second PIN wrong

exception case 7b: third PIN wrong

(Ludewig and Lichter, 2013; V-Modell XT, 2006)

Use Case Diagrams

Example: Use Case Diagram of the ATM Use Case
Use Case Example

```
name: Authentication
goal: the client wants access to the ATM
pre-condition: the ATM is operational, the welcome screen is displayed, card and PIN of client are available
post-condition: client accepted, services of ATM are offered
post-cond. in exceptional case: access denied, card returned or withheld, welcome screen displayed

actors:
client (main actor), bank system

open questions:
none

normal case:
1. client inserts card
2. ATM reads card, sends data to bank system
3. bank system checks validity
4. ATM shows PIN screen
5. client enters PIN
6. ATM reads PIN, sends to bank system
7. bank system checks PIN
8. ATM accepts and shows main menu

exception case:
2a. card not readable
   2a.1 ATM displays "card not readable"
   2a.2 ATM returns card
   2a.3 ATM shows welcome screen

2b. card readable, but not ATM card

2c. no connection to bank system

3a. card not valid or disabled

5a. client cancels

5b. client doesn't react within 5 s

6a. no connection to bank system

7a. first or second PIN wrong

7b. third PIN wrong

(Ludewig and Lichter, 2013; V-Modell XT, 2006)
```
Message Sequence Charts, ITU standardized in different versions (ITU Z.120, 1st edition: 1993); often accused of lacking a formal semantics.

Sequence Diagrams of UML 1.x (one of three main authors: I. Jacobson)

SDs of UML 2.x address some issues, yet the standard exhibits unclarities and even contradictions (Harel and Maoz, 2007; Störrle, 2003).

For the lecture, we consider Live Sequence Charts (LSCs) (Damm and Harel, 2001; Klose, 2003; Harel and Marelly, 2003), who have a common fragment with UML 2.x SDs (Harel and Maoz, 2007).

\[\begin{align*}
\text{SD Body Building Blocks} \\
\text{LSC Body: Abstract Syntax}
\end{align*}\]
Can be used to precisely describe the interactions of a use-case.

Note that messages in a chart are cyclic if messages in a chart are not well-formed, so such diagrams do not exist as a partial order. If messages in a chart are well-formed, then there doesn't exist a partial order so such diagrams, i.e., visual diagrams, can be used to precisely define syntax, weak point, strong point.

For each location \( l \in L \), \( l \sim l \), \( l \prec l \) are local invariants, \( l \sim l \), \( l \prec l \) are weak points, \( l \sim l \), \( l \prec l \) are strong points.

References


