Softwaretechnik / Software-Engineering

Lecture 8: Use Cases and Scenarios

2016-06-02

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Topic Area Requirements Engineering: Content

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

- User Stories
- Use Cases
  - Use Case Diagrams
- Sequence Diagrams
  - A Brief History
  - Live Sequence Charts
    - Syntax:
      - Elements, Locations,
    - Semantics:
      - Cuts
      - Firedsets,
      - Automaton Construction
Scenarios
One quite effective approach:

try to **approximate** the requirements with positive and negative **scenarios**.

- Dear customer, please describe example usages of the desired system.
  
  Customer intuition: **“If the system is not at all able to do this, then it’s not what I want.”**

- Dear customer, please describe behaviour that the desired system must not show.
  
  Customer intuition: **“If the system does this, then it’s not what I want.”**

- From there on, refine and generalise:
  
  what about exceptional cases? what about corner-cases? etc.

- Prominent early advocate: **OOSE** *(Jacobson, 1992).*
Example: Vending Machine

- **Positive scenario**: Buy a Softdrink
  1. Insert one 1 euro coin.
  2. Press the ‘softdrink’ button.
  3. Get a softdrink.

- **Positive scenario**: Get Change
  1. Insert one 50 cent and one 1 euro coin.
  2. Press the ‘softdrink’ button.
  3. Get a softdrink.
  4. Get 50 cent change.

- **Negative scenario**: A Drink for Free
  1. Insert one 1 euro coin.
  2. Press the ‘softdrink’ button.
  3. Do not insert any more money.
  4. Get two softdrinks.
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
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 realised.

**Proposed card layout** (front side):

<table>
<thead>
<tr>
<th>priority</th>
<th>unique identifier, name</th>
<th>estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>As a [role] I want [something] so that [benefit].</td>
<td></td>
</tr>
<tr>
<td>risk</td>
<td>real effort</td>
<td></td>
</tr>
</tbody>
</table>
**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

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

*(Rupp and die SOPHISTen, 2009)*

**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).

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

| risk | real effort |
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 Cases
**Use Case: Definition**

**use case** – 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 and 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

**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  

| exc. case 2b | card readable, but not ATM card  
| exc. case 2c | no connection to bank system  
| exc. case 3a | card not valid or disabled  
| exc. case 5a | client cancels  
| exc. case 5b | client doesn’t react within 5 s  
| exc. case 6a | no connection to bank system  
| exc. case 7a | first or second PIN wrong  
| exc. case 7b | third PIN wrong  

(Ludewig and Lichter, 2013; V-Modell XT, 2006)
Use Case Diagrams
Use Case Diagrams: Basic Building Blocks

- Actor
- Use Case
- Actor participates in use case

or:

(use case name)
### Use Case Example: Authentication

<table>
<thead>
<tr>
<th>name</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>goal</td>
<td>the client wants access to the ATM</td>
</tr>
<tr>
<td>pre-condition</td>
<td>the ATM is operational, the welcome screen is displayed, card and PIN of client are available</td>
</tr>
<tr>
<td>post-condition</td>
<td>client accepted, services of ATM are offered</td>
</tr>
<tr>
<td>post-cond. in exceptional case</td>
<td>access denied, card returned or withheld, welcome screen displayed</td>
</tr>
<tr>
<td>actors</td>
<td>client (main actor), bank system</td>
</tr>
<tr>
<td>open questions</td>
<td>none</td>
</tr>
</tbody>
</table>

#### 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

#### exc. case 2b: card readable, but not ATM card
- no connection to bank system
- card not valid or disabled
- client cancels
- client doesn't react within 5 s
- no connection to bank system
- first or second PIN wrong
- third PIN wrong

(Ludewig and Lichter, 2013; V-Modell XT, 2006)
Use Case Example: Authentication

**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
1. ATM displays “card not readable”
2. ATM returns card
3. ATM shows welcome screen

---

Ludewig and Lichter, 2013; V-Modell XT, 2006
Use Case Diagrams: More Building Blocks

More notation:

⟨actor name⟩ → 〈use case name〉
or:

〈use case name〉

More notation:

a.b.c.d.e:

⟨after⟩

use case A

⟨(extends)⟩

“procedure call”

x.y.z:

use case B

a.b.c.x.y.z.d.e

a.b.c.d.e

use case A

⟨(uses)⟩ or ⟨(include)⟩

use case B

b.d
Use Case Diagram: Bigger Examples

(Ludewig and Lichter, 2013)
Use Case Diagram: Bigger Examples

Survey of Use Cases

- Administrator
  - Administrating Server
  - Administrating User

- Applicant
  - Indicating Assistance
  - Administrating Registration
  - Log-On/Log-Off

- Examiner
  - Informing
  - Examining Registration
  - Forwarding DPMA Application

- DPMA
  - Printing Registration
  - Preparing Registration
  - Processing Registration
  - Completing Registration
  - Generating Registration for DPMA
  - Printing/Faxing DPMA Registration

(V-Modell XT, 2006)
Content

- User Stories
- Use Cases
  - Use Case Diagrams
- Sequence Diagrams
  - A Brief History
  - Live Sequence Charts
    - Syntax:
      - Elements, Locations,
    - Semantics:
      - Cuts
      - Firedsets,
      - Automaton Construction

Info III
Sequence Diagrams
• **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)
Live Sequence Charts: Syntax (Body)
LSC Body Building Blocks

- instance line head
- (cold) local invariant
- (hot) line segment
- coregion
- simultaneous region
- (cold) line segment
- instantaneous message
- (hot) condition
- asynchronous message
The Plan: A Formal Semantics for a Visual Formalism

concrete syntax
(diagram)

abstract syntax

$\left( L, \preceq, \sim, I, Msg, Cond, LocInv, \Theta \right)$

semantics
(Büchi automaton)
Definition. [LSC Body]
Let $\mathcal{E}$ be a set of events and $\mathcal{C}$ a set of atomic propositions, $\mathcal{E} \cap \mathcal{C} = \emptyset$.
An LSC body over $\mathcal{E}$ and $\mathcal{C}$ is a tuple

$((L, \preceq, \sim), I, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)$

where

- $L$ is a finite, non-empty of locations with
  - a partial order $\preceq \subseteq L \times L$,
  - a symmetric simultaneity relation $\sim \subseteq L \times L$ disjoint with $\preceq$, i.e. $\preceq \cap \sim = \emptyset$,
- $I = \{I_1, \ldots, I_n\}$ is a partitioning of $L$; elements of $I$ are called instance line,
- $\text{Msg} \subseteq L \times \mathcal{E} \times L$ is a set of messages with $(l, E, l') \in \text{Msg}$ only if $(l, l') \in \prec \cup \sim$; message $(l, E, l')$ is called instantaneous iff $l \sim l'$ and asynchronous otherwise,
- $\text{Cond} \subseteq (2^L \setminus \emptyset) \times \Phi(\mathcal{C})$ is a set of conditions with $(L, \phi) \in \text{Cond}$ only if $l \sim l'$ for all $l \neq l' \in L$,
- $\text{LocInv} \subseteq L \times \{\circ, \bullet\} \times \Phi(\mathcal{C}) \times L \times \{\circ, \bullet\}$ is a set of local invariants with $(l, l', l, l') \in \text{LocInv}$ only if $l \prec l'$, $\circ$: exclusive, $\bullet$: inclusive,
- $\Theta : L \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \rightarrow \{\text{hot}, \text{cold}\}$ assigns to each location and each element a temperature.
• locations $\mathcal{L}$,
• $\preceq \subseteq \mathcal{L} \times \mathcal{L}, \sim \subseteq \mathcal{L} \times \mathcal{L}$
• $\mathcal{I} = \{I_1, \ldots, I_n\}$,
• $\text{Msg} \subseteq \mathcal{L} \times \mathcal{E} \times \mathcal{L}$,
• $\text{Cond} \subseteq (2^\mathcal{L} \setminus \emptyset) \times \Phi(C)$
• $\text{LocInv} \subseteq \mathcal{L} \times \{\circ, \bullet\} \times \Phi(C) \times \mathcal{L} \times \{\circ, \bullet\}$,
• $\Theta : \mathcal{L} \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \rightarrow \{\text{hot}, \text{cold}\}$. 
locations $L$,

$\leq \subseteq L \times L$, $\sim \subseteq L \times L$

$\mathcal{I} = \{I_1, \ldots, I_n\}$,

$\text{Msg} \subseteq L \times E \times L$,

$\text{Cond} \subseteq (2^L \setminus \emptyset) \times \Phi(C)$

$\text{LocInv} \subseteq L \times \{\circ, \bullet\} \times \Phi(C) \times L \times \{\circ, \bullet\}$

$\Theta : L \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \rightarrow \{\text{hot, cold}\}$.

$L = \{e_{1,0}, \ldots, e_{1,4}, e_{2,1}, e_{12} < e_{13}, e_{12} < e_{14}, e_{13} + e_{14}, e_{14} + e_{13}, \ldots, e_{22} \sim e_{31}, e_{1d} \sim e_{20}, \ldots\}$

$\text{Msg} = \{(e_{11}, A, e_{21}), \ldots\}$

$\text{Cond} = \{(s(e_{22}), c_2 < c_3)\}$

$\text{LocInv} = \{(e_{1}, 0, c_1, e_{1}, \circ)\}$

$\Theta : e_{10} \mapsto \text{hot}$

$e_{20} \mapsto \text{cold}$

$(e_{1}, A, e_{21}) \mapsto \text{hot}$

$s \mapsto \text{not}$

$\lambda \mapsto \text{cold}$
Concrete vs. Abstract Syntax

- \( \mathcal{L} : l_{1,0} \prec l_{1,1} \prec l_{1,2} \prec l_{1,3}, \ l_{1,2} \prec l_{1,4}, \ l_{2,0} \prec l_{2,1} \prec l_{2,2} \prec l_{2,3}, \ l_{3,0} \prec l_{3,1} \prec l_{3,2}, \ l_{1,1} \prec l_{2,1}, \ l_{2,2} \prec l_{1,2}, \ l_{2,3} \prec l_{1,3}, \ l_{3,2} \prec l_{1,4}, \ l_{2,2} \sim l_{3,1}, \)
- \( \mathcal{I} = \{\{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}\}, \{l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}\}, \{l_{3,0}, l_{3,1}, l_{3,2}\}\}, \)
- \( \text{Msg} = \{(l_{1,1}, A, l_{2,1}), (l_{2,2}, B, l_{1,2}), (l_{2,2}, C, l_{3,1}), (l_{2,3}, D, l_{1,3}), (l_{3,2}, E, l_{1,4})\} \)
- \( \text{Cond} = \{\{l_{2,2}\}, c_2 \land c_3\}\),
- \( \text{LocInv} = \{(l_{1,1}, c_1, l_{1,2}, \bullet)\}\)
Well-Formedness

**Bondedness/no floating conditions:** (could be relaxed a little if we wanted to)

- For each location \( l \in L \), *if* \( l \) is the location of
  - a **condition**, i.e. \( \exists (L, \phi) \in \text{Cond} : l \in L \), or
  - a **local invariant**, i.e. \( \exists (l_1, \iota_1, \phi, l_2, \iota_2) \in \text{LocInv} : l \in \{l_1, l_2\} \),

then there is a location \( l' \) **simultaneous** to \( l \), i.e. \( l \sim l' \), which is the location of

- an **instance head**, i.e. \( l' \) is minimal wrt. \( \preceq \), or
- a **message**, i.e.

\[
\exists (l_1, E, l_2) \in \text{Msg} : l \in \{l_1, l_2\}.
\]

**Note:** if messages in a chart are **cyclic**, then there doesn’t exist a partial order (so such diagrams **don’t even have** an abstract syntax).
Tell Them What You’ve Told Them...

- **User Stories**: simple example of scenarios
  - **strong point**: naming tests is necessary,
  - **weak point**: hard to keep overview; global restrictions.

- **Use-Cases**:
  - interactions between system and actors,
  - be sure to elaborate exceptions and corner cases,
  - in particular effective with customers lacking technical background.

- **Use-Case Diagrams**:
  - visualise which participants are relevant for which use-case,
  - are rather **useless** without the underlying use-case.

- **Sequence Diagrams**:
  - a **visual formalism** for interactions, i.e.,
    - precisely defined syntax,
    - precisely defined semantics (→ next lecture).
  - Can be used to precisely describe the interactions of a **use-case**.
References
References


