Softwaretechnik / Software-Engineering

Lecture 08: Scenarios and Use Cases

2015-06-08

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal
Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:
• Consistency, Completeness, etc. for Decision Tables.

This Lecture:
• Educational Objectives:
  • What is a scenario/an anti-scenario?
  • What is included in a use case? In a use case diagram?
  • What is the abstract syntax of this Live Sequence Chart (LSC)?
  • Which are the cuts and fired sets of this LSC?
  • Construct the TBA of a given LSC body.
  • Given a set of LSCs, which scenario/anti-scenario/requirement is formalised by them?
  • Formalise this positive scenario/anti-scenario/requirement using LSCs.

Content:
• Scenarios in Requirements Engineering
• User Stories; Use Cases and Diagrams
• Live Sequence Charts

You Are Here

Introduction

Development Process, Metrics

Requirements Engineering

Architecture & Design

Construction Models

Testing, Formal Verification

Sealed Take

Wrap-Up
Example: V ending Machine

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

Negative scenario:
(i) After switching on, insert no money.
(ii) Press the 'tea' button.
(iii) Get a tea.
(iv) Get 100 cent change.

Notations for Scenarios

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

First prominent recognition: OOSE (Jacobson, 1992)

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

"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, 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) — how to tell whether the user story has been realised.

User Stories: Discussion

Proposed card layout (front side):

priority, unique identifier, name
estimation
As a [role] I want [something] so that [benefit].
risk
real effort
easy to create
close contact to customer
customers are usually not trained in writing requirements
may get difficult to keep overview over whole system to be developed
strong dependency on competent developers
estimation of effort may be difficult
not easy to cover non-functional requirements and restrictions (Balzert, 2009)

Recall:

Natural Language Patterns

Natural language requirements can be written using A, B, C, D, E, F, 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) (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).
Use Case: Definition

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)

- **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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Goal</th>
<th>Pre-condition</th>
<th>Post-condition</th>
<th>Post-cond. in exceptional case</th>
<th>Actors</th>
<th>Open questions</th>
<th>Normal case</th>
<th>Exception case 2a</th>
<th>Exception case 2b</th>
<th>Exception case 3a</th>
<th>Exception case 5a</th>
<th>Exception case 5b</th>
<th>Exception case 6a</th>
<th>Exception case 7a</th>
<th>Exception case 7b</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authentication</strong></td>
<td>the client wants access to the ATM</td>
<td>the ATM is operational, the welcome screen is displayed, card and PIN of client are available</td>
<td>client accepted, services of ATM are offered</td>
<td>access denied, card returned or withheld, welcome screen displayed</td>
<td>client (main actor), bank system</td>
<td>none</td>
<td>1. client inserts card 2. ATM read 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</td>
<td>2a.1 ATM displays “card not readable” 2a.2 ATM returns card 2a.3 ATM shows welcome screen</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Providing an example using open questions)

**Use Case Diagrams**

[Diagram of use case example]
**Use Case Diagrams**

- Authenticate bank system

More notation:
- use case A extends use case B
- use case A uses use case B
- or use case A includes use case B

**Use Case Diagram: Bigger Examples**

ATM info services
- print statement [not auth.]
- query balance [print statement]
- transactions
- get cash
- define standing order

authentication
- extends
- include
- include
- include
- extends

(Ludewig and Lichter, 2013)

**V-Modell XT, 2006**

**Sequence Diagrams**

Recall: V ending Machine Example

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

- Negative scenario:
  1. After switching on, insert no money.
  2. Press the 'tea' button.
  3. Get a tea.
  4. Get 100 e change.

**Simple Notation**

-understandable by customer
-precision complexity of definition
-natural language feels like ++

-visual formalism
-logical (temporal) logic

**History**

Most Prominent: Sequence Diagrams — with long history:

- Message Sequence Charts, standardized by the ITU in different versions (ITU Z.120, 1stedition: 1993), often accused to lack a formal semantics.

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

Most severe drawbacks of these formalisms:

- unclear interpretation: example scenario or invariant?
- unclear activation: what triggers the requirement?
- unclear progress: must all messages be observed?
- conditions merely comments
- no means (in language) to express forbidden scenarios.


