Scenarios and Use Cases

Daniel M. Berry

Using a Computer-Based System

Consider an interactive computer-based system (CBS), *S*. It is called interactive because its users interact with it.

A user tends to think of *S* in the ways that he or she will use it.

Use Case

Each such particular way to use *S* is called a *use case*.

It is one case of the many ways to use *S*.

A use case (UC) is expressed in natural language as a simple imperative sentence, e.g.:

- **Insert a coin into the coinSlot**
- **Push and walk through the barrier**

Scenario

A UC should not be confused with a closely related concept called a *scenario*. A scenario of *S* is a particular sequence of interaction steps between a user of *S* and *S*.

Use Cases and Scenarios

The relation between UCs and scenarios can make both terms clearer.

A single use case contains many, many scenarios.

A UC *U* of *S* has a so-called *typical* scenario. This scenario is that identified by the stakeholders of *S* as being the normal case of *U* that proceeds with all decisions being made in the so-called normal or typical way.

Typical Scenarios for UCs

For UC: Insert a coin into the coinSlot

1. **The visitor is able to drop a coint into the coinSlot**
2. **The coin is accepted by the coinSlot**
3. **The coinSlot informs the turnstileSystem that someone paid**

Typical Scenarios for UCs

For UC: Push and walk through the barrier

1. **The visitor places both hands on the barrier**
2. **The visitor applies horizontal force to the barrier**
3. **The barrier starts to move**
4. **The vistor walks through the barrier**

Variations

A UC consists also of variations called *alternatives* and *exceptions*.

Alternatives (As)

An alternative of UC *U* is a sub-use-case that achieves the main goal of *U* through different sequences of steps or fails to achieve the goals of *U* although it follows most of the steps of *U*.

Exceptions (Es)

An exception of UC *U* is a sub-use-case that deals with the conditions along the typical scenario and other sub-use-cases that differ from the norm and those already covered.

As vs Es

The distinction between alternatives and exceptions is not really important, …

so long as you find all of *both* of them.

So think of them as prompts that help you find all of them.

As & Es for Example

For UC: Insert a coin into the coinSlot

- **Fail to insert coin into the coinSlot**
- **Insert a coin into the coinSlot that is returned unused**
- **Hit the coinSlot in anger**
- **Curse the coinSlot in anger**

As & Es for Example

For UC: Push and walk through the barrier

- **Push the barrier, feel that it moves, but don’t walk through the barrier**
- **Push the barrier, feel that it moves, and walk through the barrier only part way before feeling that it locks**
- **Push the barrier and feel that it is locked**
- **Hit the barrier in anger**
- **Curse the barrier in anger**

Brainstorm to Find As & Es!

Brainstorm to find as many alternatives and exceptions as you can!

You can always throw out ones that prove useless later on.

It helps to have a twisted psyche, a devious, diabolical mind!

Heh heh heh!!

Misuse Cases

(Sindre and Opdahl) and Alexander have proposed *misuse* cases to capture use cases that a system must be protected against, that you do not want to happen at all, e.g., a security breach.

E.g., For UC: Insert a coin into the coinSlot

- **Insert a counterfeit coin into the coinSlot**

Sharing of Subsequences

Observe that all the scenarios of a use case share many subsequences of steps.

Some sets of these subsequences of steps may constitute sub-use-cases that are worth considering as use cases that can be used by other use cases.

Purpose of Ss & UCs

Thus, scenarios and use cases are means to document the way users use *S* in order to do their work.

Scenarios and use cases of *S* help to identify requirements of *S* in the sense that *S* must be able to do the system's part of every scenario and instantiated use case.

Elicitation of Ss & UCs

Scenarios and use cases are considered good vehicles for getting users to describe how they use or will use systems.

Observation of Ss & UCs

Scenarios and use cases can be inferred by requirements engineers from their in situ observations of the work place.

Other Uses of Ss & UCs

Scenarios and use cases of *S* form a good basis for writing user's manuals and on-line help about *S* since all must describe how the users use *S*.

Scenarios and use cases of *S* form a good basis for building covering test cases for testing *S* in the sense that all must cover all the ways the users will use *S*.

To An Example

Now let us identify the use cases and some scenarios for the CBS whose Domain Model we built in class.

Feedback

As you are adding UCs to a growing list of UCs for a CBS, you will occasionally find the need to modify your DM of the CBS to be consistent with your Ss & UCs, …

and then vice versa, …

and then vice versa, …

Review of Ss & UCs

We have seen how Ss & UCs can help in identifying requirements, i.e., in requirements elicitation.

They give something concrete that the users and clients can latch on to to see how the proposed system will interact with them.

Ss & UCs Useful Otherwise

Ss & UCs are useful for other activities as we shall see in future lectures:

- **as a basis for user validation of requirements understanding**
- **as a basis of an active review of specifications**
- **as a basis of a quick and dirty prototype**
- **as a basis of test cases for the eventual implementation**

Wow!!

Pluses for Ss & UCs

According to Jo Atlee:

- ⊕ **Ss & UCs are simple, easy to create.**
- ⊕ **All clients understand S&Uc. Often the Ss & UCs are the only things the clients understand of the requirements document.**

Pluses, Cont'd

- **Ss & UCs usually reflect the user's essential requirements. You will want to keep the use cases in mind as the specification document is elaborated. It is easy to get carried away and populate specification with bell and whistle functionality. You want to be sure that the core functionality, captured by the use cases, is not lost.**

Pluses, Cont'd

- **Ss & UCs separate normal behaviour from exceptional behavior, making it easier to scrutinize various alternatives, since each can be considered separately.**

Negatives for Ss & UCs

According to Jo Atlee:

- ⊖ **Scenarios don’t scale well in size or complexity: As requirements and scenarios grow in size (usually exponentially), it gets harder and harder to coherently organize them. Abstracting scenarios into use cases helps a bit.**