Introduction to Use Cases

INCOSE Bristol Local Group
28th March 2007

Ian Gibson, with input from Matthew Hause (Artisan Software)
The History of Use Case Modelling

• Published by Jacobson in 1987
  – Functional specification of object-oriented systems.

• Originally used to model telephony systems.

• Part of the Unified Modelling Language.
What is a Use Case?

- **The clue is in the name:**
  - A Use Case is a particular example (a case) of how a system is used, literally a “case of system use”.

- **Two formal definitions:**
  - “The Use Case construct is used to define the behaviour of a system or other semantic entity without revealing the entity’s internal structure. Each Use Case specifies a sequence of actions, including variants, that the entity can perform, interacting with actors of the entity” [OMG,2001].
  - “Use Cases record a contract between system stakeholders about the behaviours of the system under discussion under various circumstances, organized by goals of selected actors” [COCKBURN, 2000].
• A Use Case describes a set of interactions between an actor and the system of interest, in order to achieve a particular goal, prompted by some kind of triggering event.
  – A Use Case meets a need or solves a problem for an actor.

• So, what is an Actor?
  • An Actor is an entity which is eternal to the system, which interacts with the system, but is not a part of the system.
  • Actors should reflect the entire system lifecycle including: manufacture, test, installation, maintenance, etc.

• A set of Use Cases can be developed to tell “stories” of how a system will be used, from the differing points of view of each of the actors.
What is an Actor?

**Expected Actors**
- A Person
- An External System
- Abstract such as Time

**Unexpected Actors**
- Unauthorized Users
- Threats
- Adverse Environmental Conditions
• The first step in discovering a set of actors is to define a system boundary. You can then think about direct interactions and indirect interactions that will occur at this system boundary. Try to think of actors as roles not distinct people or entities.

```
class Actors

Operator
Collaborating System

Boundary of the System of Interest

Project Manager
System Acceptance Authority

Time Customer Tester Trainer Maintainer
```

“What’s my motivation?”
• When describing a Use Case, it is usual to start off by describing a “happy day” scenario. This is literally a description of what happens when everything goes as expected from the trigger event through to the final step.
  – As a general rule, all use cases should have a well defined happy day scenario.
  – The “happy day” scenario is accompanied by definitions of pre-conditions and post-conditions, which are statements which hold true before the use case starts, and after it has completed.
Use case description typical format

• For an IT system example:
  1. The <actor> chooses an option from some predefined set of things that they are allowed to do.
  2. The <system> provides the <actor> with <form XYZ>.
  3. The <actor> inputs data using a mixture of free text and selections from predefined choices (see definition of <form XYZ>.
  4. The <actor> chooses to submit the data.
  5. The <system> validates the inputs using <rule 123>.
  6. The <system> informs the <actor> that validation was successful, and provides an option for the <actor> to acknowledge this.
  7. The <actor> acknowledges that validation was successful.
  8. The <system> commits the data to <database ABC>.
• It is often the case that there will be alternative paths through a use case, depending upon decisions made by the actors, or functionality required of the system.

• These are captured as “alternative flows” and referenced to the step in the use case where they occur.

• If a similar alternative flow keeps on cropping up across several use case descriptions then it may be worth splitting that behaviour out into its own use case.
Sometimes things go wrong

- Life is not all about happiness, and neither are use cases. There will be times when the use case has to depart from the happy day scenario due to something going amiss.

- These are captured as “exceptions” and referenced to the step in the use case where they occur in a similar way to a use case.

- In today’s workshop we will tend to focus on the main “happy day” scenarios, but when developing use cases you should always ask “can this be done differently?” and “what can go wrong?”. 
Typical Use Case Template

• Name
• Purpose/Intent
• Actors involved
• Pre conditions
• Post conditions
• Triggering event
• Main flow ("happy day" scenario)
• Alternative flows
• Exceptions
• Remarks
  – These could be assumptions, constraints or system design implications.
• References to source material
• The purpose of Use Case diagrams is to show which actors are involved in the behaviour captured in each use case.
  – If this needs multiple diagrams to show all of the combinations of actors and use cases then so be it, there is no magic 7±2 rule and the structure is flat not hierarchical.
  – A secondary purpose of use case diagrams is to show any common behaviour that is shared by several use cases and any departures from the main scenario that are so complex as to warrant their own use case.
    • This is shown using «includes» and «extends» relationships.
    • The process of pulling out common behaviour and complex departures is known as “refactoring”.

© HI-Q SYSTEMS LIMITED 2007
Examples of use case diagrams

E.g. “Open document in read only mode”

E.g. “Open document in edit mode”

E.g. “Print document”

E.g. “Perform spell check on document and provide change suggestions to user”
“When I use a word, it means just what I choose it to mean – neither more nor less” *(H. Dumpty)*

- When developing use cases it is vital to define the terminology being used at the same time.
- In most Use Case/UML methods this is known as the “domain model”.
- Defining key terminology and the relationships that exist allows the analysts to maintain coherence across the use cases, whilst embodying stakeholder language.
The domain model as a UML class diagram

- Class3 is a kind of Class1
  E.g. Ferrari is a kind of car

- Class3 is associated with Class4
  E.g. Ferraris are made by FIAT

- Class1 has Class2
  E.g. A car has wheels

- E.g. A car has wheels
So what does this have to do with requirements?

- **Use cases** describe the way in which a system can be expected to behave in response to a set of stimuli, from a range of actors, in a range of initial conditions and operational environments.
- Each statement about something the system does can be considered as a candidate requirement.
  - These will often overlap as the same rules about system functionality will tend to hold true across a set of use cases.
- Each statement about something an actor does provides an implicit interface requirement.
- Each pre/post condition provides information about potential system states.
- Each constraint identified during the use case analysis can be considered as a candidate non-functional requirement.
Validating use cases with stakeholders

• Use cases can be used to validate the required system behaviour with stakeholders, by providing them with a narrative that incorporates aspects of the operational environment.
• The narrative approach allows you to ask them if this is the way that the actors would do things, and to ensure that the system responses are appropriate to the operational task.
• You can often elicit additional assumptions and requirements by walking a stakeholder through the steps in the use case.
• There are three UML diagrams that can be particularly useful when elaborating use cases:
  – Class diagram (to capture a static view of the collaborating actors and system elements);
  – Activity diagram;
  – Sequence diagram.

• For those of you using DODAF/MODAF, the OV-1, OV-5, OV-6c and SV-10c cover similar ground.
Use cases in summary

- Capture a set of stories about how actors interact with a system of interest.
- Can be elaborated using activity diagrams and sequence diagrams.
- Use case diagrams are used to show the involvement of actors in use cases, and how use cases may call up other use cases for common or exceptional behaviour.
- Use case diagrams are not supposed to show hierarchical decomposition, nor are they used to represent functions and data flows between them.
- Use case descriptions can be used to drive out implicit behavioural rules that apply across many operational situations – candidates for requirements.
Plan for today’s tutorial

- Read background material.
- Establish a set of stakeholders that will be the basis of our actors.
  - Supplement this with any actors that may not be stakeholders (such as collaborating systems).
- Develop an initial set of categories (system key features) that we may wish to explore using use cases.
- Use these actors and categories to derive some questions for the stakeholders about their goals and needs.
- Define a set of use cases describing the behaviour that is necessary to meet these goals and needs.
  - Go back to the stakeholders for clarification as necessary.
  - Define domain terminology and relationships as you go.
- Derive functional and non-functional requirements based upon the use cases descriptions and pre/post conditions.
- Review the outputs for coherence and consistency.
- Review the session and prepare feedback.
How the group found it…
Strengths

• Provides a sequential logical method.
• Template is useful to provide structure to the analysis.
• Forces you to focus on system boundaries and stakeholder viewpoints.
• Exposes questions that lead to new requirements, identifying inconsistencies and assumptions.
• Provides a mechanism for buying off stakeholder requirements.
• Can fit in with other tools and techniques.
Issues

• Easy to get drawn into the “solution space”
  – Need to understand where the use case is pitched within the system of systems.
  – Need to understand the system boundaries.

• You could potentially waste time and effort analysing at a lower level before adequately agreeing the requirements.
  – This can be mitigated by enforcing review points once a set of candidate use cases exists.
  – It may be necessary to prioritise effort on those use cases that will bring the most benefit (e.g. improve your understanding of the system of interest), rather than simply starting at one end and working through the whole set.

• Use cases are not stand alone.
Usefulness

• The technique looks like it should be useful, but would need further experience to feel happy using it.
Example Use Case Diagram 1: Congestion Charging Scheme Finance Context
Example Use Case Diagram 2: Payment Subsystem Finance Context

Motorist

- Pre-pay charges
- Pay charges online
- Pay charges by post
- Pay charges by mobile

This actor could be replaced by the more generic "Road User" actor.
Example Domain Diagram Fragment: Relationships between some key actors