Discovering Use Cases

The Project Process

In creating a project most books on the subject split the process into four stages

- Inception - initial planning
- Elaboration - refining the design
- Construction - building the system
- Transition - installation support

Inception

Inception is the phase of the project where things come together in their early form.

- What is the system going to do?
- What is the business case for the system?
- Are we better off buying a system off the shelf?

We will create such things as

- Project specification – containing a written specification of the what and why of the system
- Ethical review – what impact will the system have from an ethical stand point?
- Event tables – detailing the events within the system and who is triggering them
- Initial use case diagram + descriptions – detailing who is going to use the system and the tasks they will perform
- Class diagram – indicating the classes that will be created within the system along with their attributes/properties & operations/methods
- Early prototype – a “smoke and mirrors” model of how the system will work
- Test plans – based on input fields we will draw up test data for all parts of the system
- Usability testing – we will test the prototype with potential users to see if the design works or not

Elaboration

The elaboration phase is the stage in the system where we really start to fine tune the use cases and classes for the system.

In this phase we will look at how the system work in a dynamic way using sequence diagrams.

We will also look at how we are going to store the data using entity relationship modelling.

Construction

This is the stage where we start building the tables, stored procedures and classes that make the system work.

Transition

This is the stage where the system is delivered and installed. We deal with support for the end user and any technical issues related to installation.

Dealing with Uncertainty

The big lie at the heart of many of these phases of software development is that it is a nice tidy sequential process. On paper it all looks so clean and tidy but in reality this is never the case.

Proofs of Concept

What you will find in developing a large system is that the process of creating the system starts to raise other questions.

- Should the main screen be blue or green?
- Will the language I am using work on that version of the operating system?
- How do I process this list of data to format it in a specific way?
- I am really not very clear on what the user wants with this single requirement?
- Is it possible to connect this system to Oracle rather than SQL server?

Questions such as these (and lots of others) will inevitably crop up as you create the system.

The way to handle many of these issues is to create smaller sub systems as a proof of concept.

The nice thing about the sub systems is that they may be knocked together quite quickly with the option of ignoring some elements of good practice.

Should the main screen be blue or green?

Create two mock up systems and see what the users think.

Will the language I am using work on that version of the operating system?

Create a small program using the technologies for the main one and test it.

How do I process this list of data to format it in a specific way?

Create a small program that handles this task and this one only. Figure out how this might work and then integrate into the main system.

I am really not very clear on what the user wants with this single requirement?

Arrange a meeting with the user to discuss in detail this requirement. Use prototypes if required so that they may visualise possible solutions.

Is it possible to connect this system to Oracle rather than SQL server?

Write some code to test the connectivity. Again integrate when ready into the main system

If in doubt ask!

Digesting the Problem

The Specification

Last week we started to think about the specification for The Project Bank.

You were given the initial seed of the idea - the ability to record data from business cards and flyers.

> *As a self employed computing consultant you encounter a large number of people whilst involved in your networking activities. One consequence of these activities is that you accumulate a huge number of business cards/flyers. These documents contain a large quantity of useful information that needs to be input into suitable database system.*

For this example we shall work through some steps allowing us to really think about what is going on with this specification. At the end of this we shall produce a use case diagram in Enterprise Architect.

Imagine Yourself Doing the Task

One common problem with initial specification such as this is that they are often too vague to really tell use what is going on.

Although it isn't necessary to produce a formal role-play it is a good idea to place oneself in the situation of the potential user.

- Who
- What
- When
- Where
- Why

(If you are not clear on any of these points then you must return to the client and find out.)

Ok – let's set the scene.

You are a self employed consultant going to network events and meeting people on a one to one basis (perhaps on a train).

As a normal part of your interaction you exchange business cards.

OK – what happens to the business card?

It is unlikely that you process the business card there and then.

That would be a matter of saying "hold on a minute mate, just sit and wait there till I update my database".

It is far more likely that you stick the card in your bag or pocket and carry on the conversation.

So what next?

Do you wait for a spare moment till you update the database?

Possibly – you could have a phone app that allows you to do this when you have a spare moment on the train!

Or what is more likely is that at some point when you get home you realise that you have accumulated loads of cards in your bag / pockets.

So do you deal with them then?

Probably not! The odds are that you have a box which you shove them into until there is no more room in the box and you now have to do something with the backlog.

So now you are about to enter the data from your business cards. You make a cup of tea and make sure you have a "quiet moment" when you may sit and enter the data.

You are now faced with say 200 business cards to enter onto the system.

What is the best order for the data entry fields on the screen?

What happens if the business card's details don't match the details on the data entry screen?

How do you know when you are entering a business card that you are not entering a duplicate?

Jane Bloggs XYZ Computing 07811999999 j.bloggs@xyz.co.uk	Jane Bloggs XYZ Computing 07811999999 j.bloggs@xyz.co.uk

What happens if you are entering a business card for a company already on your system but it is an additional member of staff within that company?

Jane Bloggs XYZ Computing 07811999999	Fred Smith XYZ Computing 07811999999

j.bloggs@xyz.co.uk	f.smith@xyz.co.uk

What happens if it turns out to be the same person but some important details have changed?

Jane Bloggs XYZ Computing 07811999999 j.bloggs@xyz.co.uk	Jane Bloggs XYZ Computing 07811888888 j.bloggs@xyz.co.uk

What happens when the phone rings or somebody comes to the door and there is a distraction? Is it easy to see where you were up to when you come back to the desk?

If we have a good idea of what questions we need to ask then we are in a better position to have a go at re-writing the specification such that it includes important detail.

> *The consultant sits at their desk with a stack of business cards and flyers. They pick up a business card and start to input the details into system. The first field they enter is the name of the company. While doing this the system looks up the company name to see if it is already on the system. The next field the user enters is the name of the contact. Whilst dong this, the list of contacts for that company is displayed such that if the contact is already on the system the user may move onto another business card. At this point the user should have the opportunity to update the details on the system should they note that some aspect has change e.g. email address.*

This specification is probably still not perfect but it is a huge improvement on what we had previously.

The Event Table

The next step is to create an event table as the first tool allowing us to uncover use cases.

We are looking to identify from the specification:

- The subject (who or what is involved)
- The verb (what they are doing)

- The object (what are they doing it to)
- And the response (what happens when the verb is carried out)

One issue when creating systems is the desire to rush ahead without really thinking about what we are doing.

This combined with the desire to get things right on the first attempt (due to fear) will mean that our systems fail to meet the requirements.

Once we have a written description of what the system will do we may set about writing an event table.

The event table is a useful tool for thinking about who is doing what to the system.

This may be drawn up on your own but is best drawn up with a team of other people.

In the event table we need to identify

- The subject - who or what is triggering the event
- The verb – what is being done
- The object – the thing that it is being done to
- The response – what happens as an outcome of the event

Below is a simplistic event table for the Address Book application. (It is simplistic as it probably has too much detail – it doesn't have to be right on the first attempt)

Subject	**Verb**	**Object**	**Response**
User	Views	Address List	Addresses are listed by the system
User	Filters	Address List	Address list is filtered based on pattern
User	Adds	Address	Address is added to the system
User	Updates	Address	Address is updated on the system
User	Deletes	Address	Address is deleted from the system
System	Validates	Address	Address data is accepted or error is displayed

Work with other team members and brainstorm potential candidates.

Better to have too many and remove them as you refine your understanding rather than too few and miss something important.

Here is my first stab…

> *The consultant sits at their desk with a stack of business cards and flyers. They pick up a business card and start to input the details into system. The first field they enter is the name of the company. While doing this the system looks up the company name to see if it is already on the system. The next field the user enters is the name of the contact. Whilst dong this, the list of contacts for that company is displayed such that if the contact is already on the system the user may move onto another business card. At this point the user should have the opportunity to update the details on the system should they note that some aspect has change e.g. email address.*

"consultant sits at desk"
"pick up a business card"

It is very common at this stage to identify candidate events that are in fact outside the scope of the system. But as I said it is better to identify and reject rather than miss something important.

Don't dismiss these candidate events completely though. Hidden in here is one candidate actor for the use case "consultant" and possibly a candidate class "business card"!

What about the following?
"input the details"
"enter the company name"
"enter the name of the contact"

Another issue at this point is that we might be identifying events that are in fact describing the same thing.

Let's have a first stab (there is a blank template available on the module web site)

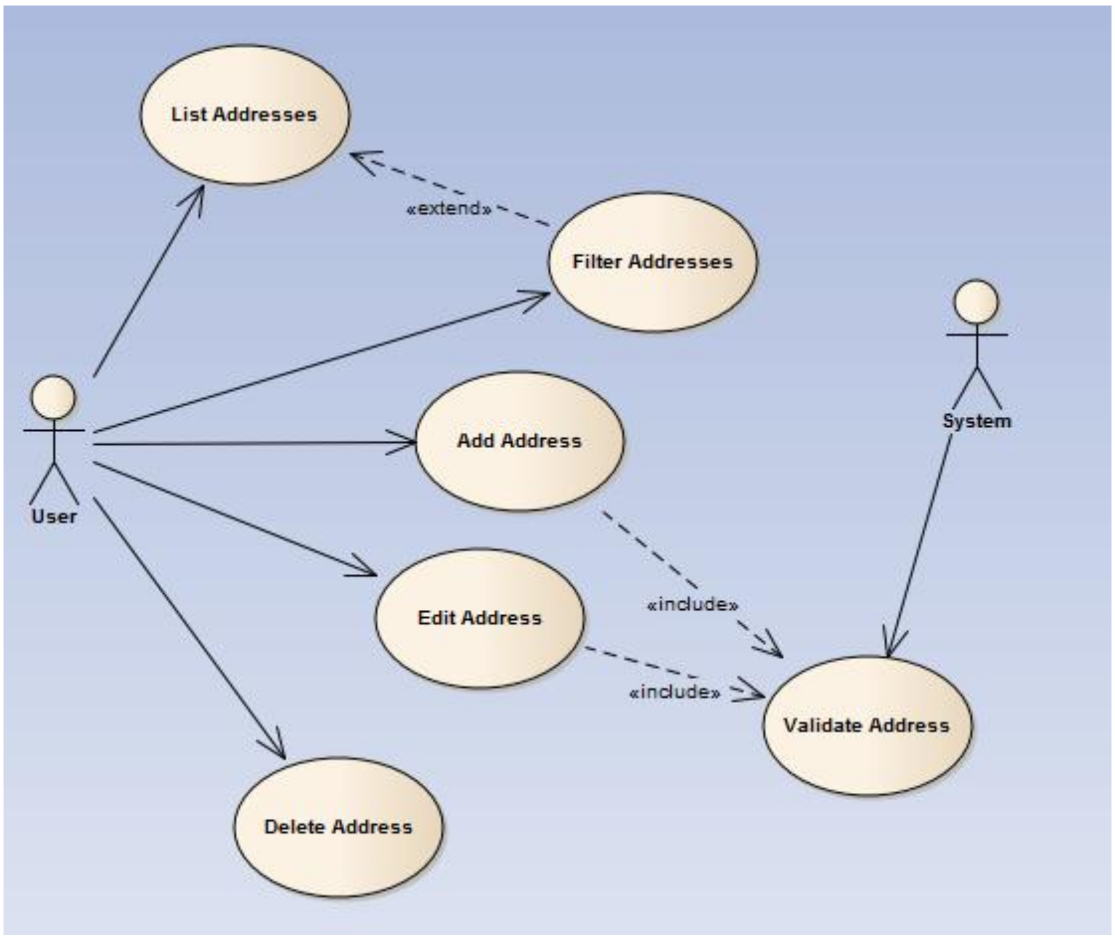
Subject	**Verb**	**Object**	**Response**
Consultant	Inputs	Card	Data accepted by the system
System	Checks	Card	Identifying duplicates
Consultant	Updates	Card	New data input

Let’s keep it fairly simple at this stage as we are more interested in working through the process rather than a perfect answer.

Initial Use Case Diagram

You should already be familiar with user case diagrams in the first year. We will explore them a bit in this module but we are more interested in seeing exactly how they turn into a real world system.

The address book use case might look something like this…



Use cases are generated from the event table.

Look at the verbs in the event tables!

Subject	**Verb**	Object	Response
User	Views	Address List	Addresses are listed by the system
User	Filters	Address List	Address list is filtered

			based on pattern
User	Adds	Address	Address is added to the system
User	Updates	Address	Address is updated on the system
User	Deletes	Address	Address is deleted from the system
System	Validates	Address	Address data is accepted or error is displayed

Actors may be generated from the subject column

Subject	**Verb**	**Object**	**Response**
User	Views	Address List	Addresses are listed by the system
User	Filters	Address List	Address list is filtered based on pattern
User	Adds	Address	Address is added to the system
User	Updates	Address	Address is updated on the system
User	Deletes	Address	Address is deleted from the system
System	Validates	Address	Address data is accepted or error is displayed

Initial Use Case Descriptions

The use case descriptions allow us to think more deeply about what is going on with the system

For example the use case description for list addresses might look like this.

Use Case Name (Short two or three word name)	List Addresses
Use Case Description (Short description)	The user views a list of addresses in the system
Use Case Author(s) (Who wrote this)	Matthew Dean
Actor(s) (Who does this)	User
Locations (Where does this happen)	On-line
Primary pathway (What is the normal "happy path" for this use case?)	List addresses User enters the system A list is displayed to the user at system start
Alternate pathways (What other paths are there that are not the "happy path"?)	There is no data in the system – an message is displayed saying so User applies filter A filtered list is displayed to the user User clears filter Full list is displayed to the user
Exception pathways (What could possibly go wrong?)	Database connection fails Error displayed to the user advising of connection problem