



March 2009

SRSM & Beyond Project

Briefing Note

An introduction to Documenting GB Smart Metering Functional Requirements with Use Cases

Version: draft_01

Alastair Manson, Engage Consulting, SRSM Project Team

An Introduction to Documenting GB Smart Metering Functional Requirements with Use Cases

Introduction

This paper discusses the SRSM approach to documenting the functional requirements for smart metering in GB.

Why the Need to Document Functional Requirements?

The SRSM project has amassed a significant body of work: the Smart Metering Operational Framework, the meter specification and the Local and WAN communications assessments. These are largely assessments of technical requirements and are without real definition of the functional requirements for smart metering (i.e. what the smart metering infrastructure will be used for).

Documented functional requirements allow a common understanding of the required behaviour of a system by a variety of people (and help to uncover where there are misaligned expectations of the eventual solution).

Documented functional requirements should help drive the development of the system and are key to testing the successful delivery of a system.

In the light of the technical descriptions described above, the functional requirements can be used to validate the technical requirements, assess the suitability of technology options and assess solutions.

There is a need to develop a common understanding of what the functionality required of smart metering for GB. This is needed so that key stakeholders have aligned expectations both at strategic and at a functional level,

What are Use Cases and Why Use This Methodology?

Why use this methodology?

Use cases are a mature methodology for discovery and capture of requirements. Although use cases may not have been widely used in the GB energy industry, they are an accepted way for functional requirements of programmes to be documented.

Use cases describe how a system behaves to provide functionality of benefit (goals) to “actors” which may be people, systems or organisations. Each use case identifies a goal and how that goal is achieved for an actor.

Use cases use natural language and put the requirements in context to allow the requirements to be easily comprehensible and communicated to a wide audience including the non technical and allow a common understanding between people with different interests and experience. For example, use cases can give members of the Marketing and IT departments of a company a shared language and common understanding of a system.

The collection of use cases for a system may be presented in a Use Case Diagram. This diagram presents a graphical overview of the functionality provided by a system in terms of actors and their goals and can provide a useful summary.

Use cases are not all the requirements of a system. They do not capture, for example, non functional requirements, data formats, user interface requirements and so on. Use cases are, however, central to the requirements and can be used to connect other requirement details. Use case development is a good way to capture information relating to other requirements – for example although data formats would be part of the technical solution, use cases can be used to capture the key information involved in an interaction.

What is the use cases methodology?

Use case modelling was originally developed in the late 1980s for software development. Since then the use case methodology has become probably the most widely used technique for capturing functional requirements of any system.

Use cases describe the behaviour of a system from a goal based perspective. What is it the system provides that is of benefit to the people (or systems) that use it? Alistair Cockburn¹ describes use cases as a contract between stakeholders (for what a system will deliver).

A use case describes the interaction between an actor and system in order to achieve a Goal. An actor may be a person, system or organisation that interacts with the system. It is worth noting that an actor is a person playing a role and the same person may play more than one role in different circumstances.

For a given system, the collection of use cases describes the functional requirements of that system.

Use cases can be written in different amounts of detail and at different levels of goal. They are intended to be a flexible tool to capture requirements accurately whilst avoiding spurious detail.

In terms of different levels, use cases can describe very high level, strategic or summary goals down to sub functions of a system. In terms of smart metering: strategic goals might be uses such as providing customer service and billing functionality without most current site visits or providing Demand Response, a functional level use case would describe scheduled read delivery or collection of ad hoc reads.

Use cases are text descriptions of system behaviour and do not require special tools, text documents following an agreed template are perfectly sufficient.

There is no single template for use case documentation, indeed, projects should be encouraged to create a template that suits their purpose. There is much agreement on the key information to be captured for a use case, although it is important to capture information appropriate to the level you are working at.

The key concepts before working on a use case are:

- System – What is the scope of the system under discussion?
- Level – What level are you looking at?

To actually document a use case you address further information that supports understanding the goal and how it is achieved as follows:

¹ Alistair Cockburn - Writing Effective Use Cases 2001

- Scope –identifies the system (the boundaries of the system) being described
- Use case (Goal) – the objective of the system that gives benefit to an actor
- Actor – anyone outside the system that interacts with the system to achieve the goal
- Stakeholder – anyone outside the system with an interest in the goal being achieved
- Primary actor – usually the actor that initiates the use case
- Preconditions and Postconditions – what must be true before and after the use case runs
(note: preconditions almost invariably point to the existence of another use case where the condition is established)
- Basic Flow - The main success scenario, a description of the steps in a straightforward case where goal is achieved.
- Extensions – Other ways in which the flow may happen.

A reasonably full use case would contain the information above.

Use cases can capture other useful information such as:

- The key information involved in an interaction
- Relationships with other use cases
- Business rules
- Frequency and performance requirements
- Assumptions
- Outstanding issues

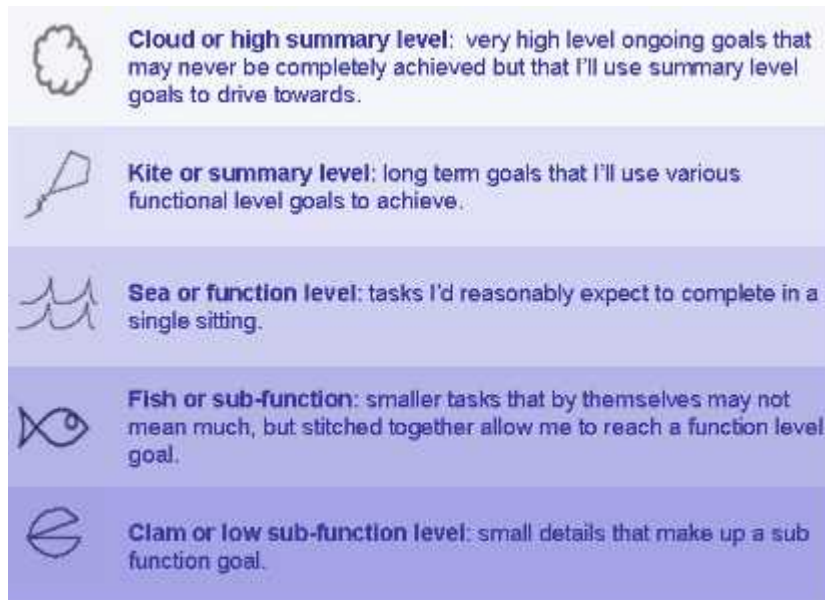
and use cases can make a useful contribution to maintaining master lists of items such as Stakeholders, Actors, Business Rules, Outstanding Issues etc.

Use cases are best developed in a collaborative way, the approach can allow different levels and functional areas to be addressed and reviewed by appropriate stakeholders, experts and users.

Levels of use case

Use cases can be written at a number of levels:

Alistair Cockburn uses a sea level analogy to describe these levels:



High Summary level use cases - provide a description of the high level goals the system is intended to deliver – the collection of high summary level use cases corresponds closely with the vision for the system.

Function level use cases – express the immediate goals actors have in using the system.

It is useful to cover the breadth of a system at high level then to drill down to a sensible level of functionality. Although use cases can be written to describe minute functionality, this is usually inappropriate detail.

The amount of detail that is recorded for a use case will vary according to the level it is written at. High summary level use cases might record a description of the goal and the stakeholders involved only – a step by step description of how it is achieved is appropriate for lower level use cases.

Use Case Diagrams

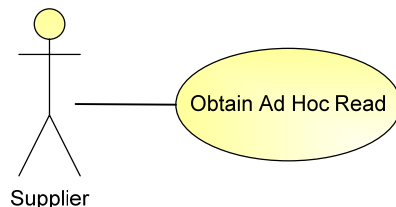
A use case diagram can be used to show actors and use cases. There is a standard notation for use cases as part of UML (the Unified Modelling Language). Actors are shown as stickmen with the name of the actor written underneath and use cases are shown as ellipses with the name of the use case written inside. The system boundary is a rectangle enclosing the use cases. Relationships are lines between actors and use cases and these may be

elaborated by arrows showing a direction of a relationship or by labelling the lines with the nature of the relationship.

There is a common misperception that the ellipses are the use cases. Use case diagrams must not be confused with use cases themselves. Use case diagrams can be considered a table of contents and although they can be a very useful way to discuss context, it is a mistake to spend too much effort elaborating relationships on use case diagrams, the real value of use cases is the text of the description.

High level use case diagrams, or lower level diagrams of functional areas may be very useful but a complex system may have dozens of use cases and although it may be possible to draw a monster spider's web diagram with all actors, use cases and relationships correctly shown, it really will serve to confuse rather than inform.

This is a simple use case diagram:



This is an example of how that use case might be documented:

Obtain Ad Hoc Read

- **Description**
This use case allows suppliers to obtain reads for specified meters outside the scheduled provision.
- **Actors**
Supplier
- **Trigger**
The Supplier realises they want the latest read from a meter (e.g. to discuss with the customer on a customer query)
- **Preconditions**
The Supplier is registered as the current supplier within the metering system (a party authorised for this transaction)
- **Post Conditions**
The Supplier has the required read
- **Basic Flow**
 1. The supplier requests the latest read of the smart metering system
 2. The smart metering system validates the request
 3. The smart metering system retrieves the latest read and sends the read to the supplier

4. The supplier receives the read

Note: this is only an example, the real use case might include more information, such as further preconditions, it would also look into what happens outside the successful flow.

Relationships between use cases

There are a number of relationships between use cases that may be used. However, as described above, extensive work elaborating relationships between use cases can be effort spent in confusing rather than elucidating requirements.

Although other relationships are possible the two most common relationships are described here:

Extends

The extends relationship extends an existing use case by defining conditions where that use case may be interrupted and an alternative scenario might happen within that existing use case. It was originally developed for coding software where a version of requirements might be “locked” and it is then realised that further behaviour needs to be described – the extended behaviour can be described in an extends use case.

We have already said we can describe extensions or alternative flows within use cases.

You can use an extends use case either as described above to avoid changing a stable use case or because it is sufficiently complicated that it warrants its own use case and it can avoid “cluttering” the original.

For example, using the Ad Hoc Read, if the validates step was found to actually need a further exchange with another party, this might be documented as an extends use case.

Includes

The includes relationship describes a normal sub routine of a base use case. This can be especially useful if the sub routine is common to a number of use cases.

Again, using the Ad Hoc Read example, if there was found to be validation routine that is common to many uses, then it could be documented using the includes relationship.

Limitations of use cases

It must be recognised that use cases are not all the requirements for a system as they don't detail user or data interfaces, data formats, algorithms etc. Specifically, use cases do not record non functional requirements.

Use cases may even make up only a small proportion of the requirements of some systems; however the use cases can be seen as a hub of requirements. While Data formats User interface and so on are not in use cases, these requirement relate to use of the system. Use cases are often seen as a project linking structure by which development can be prioritised and planned.

The process of writing use cases can help identify where there are requirements that don't "belong" in the use case and it is worth capturing and feeding these to be listed for assessment elsewhere.

General Guidelines

The key thing to remember with use cases is that they are intended to simply communicate what a system does. **It is far better to write an easy to understand description that is readable and accurate, than to fill in every part of a use case with immense detail that may be wrong.** It is very much the difference between accuracy and precision. The effort should go into defining at high level accurately, the high level use cases and the high level parts of use cases, rather than putting effort into elaborating use cases that may be wrong.

It is valuable therefore to look at the breadth of system and describe that at high level and to allow a chance for stakeholders to review and correct this view then to drill down to a lower level of detail where appropriate and repeat the process of review.

Broadly, in writing use cases, you can prioritise as follows:

1. Actors and Goals
2. Main success scenario
3. Failure conditions
4. Failure handling

Attempting to document a lower level or more detailed level before a higher one is reasonably well understood can mean working on incorrect knowledge, and can result in serious errors or holes being left in requirements.

It is also important that, as you drill down, discoveries are fed back to the appropriate level. It is quite possible, for example, to discover an actor that had not been identified previously when you are documenting how the system handles a failure.

The Approach to Use Cases for SRSM

The purpose of the exercise is to describe the functional requirements of smart metering. This needs to be done for the industry; this exercise gives a starting point.

This documentation may later be used in many ways such as:

- It may be used to help identify further requirements of smart metering such as data interfaces, user interfaces business rules and performance
- It may later be further developed to better understand the component parts of the system
- It may later be further developed to design the system or parts of it
- It may be used to develop test or evaluations

But these later uses are for consideration later and depend on programme decisions moving forward.

Smart metering is a huge system to define and it will be important that the exercise is broken into sensible chunks. These can be determined as the work progresses and can be flexible as other work is prioritised. In general though, the project will follow good practice of documenting at High Level to provide breadth of the system, then drilling down to lower levels in batches as agreed.

Use cases are best developed as a collaborative exercise, the project will use a variety of central documentation and review, brainstorming workshops, and informal and more formal one to one sessions with individual organisations as appropriate.

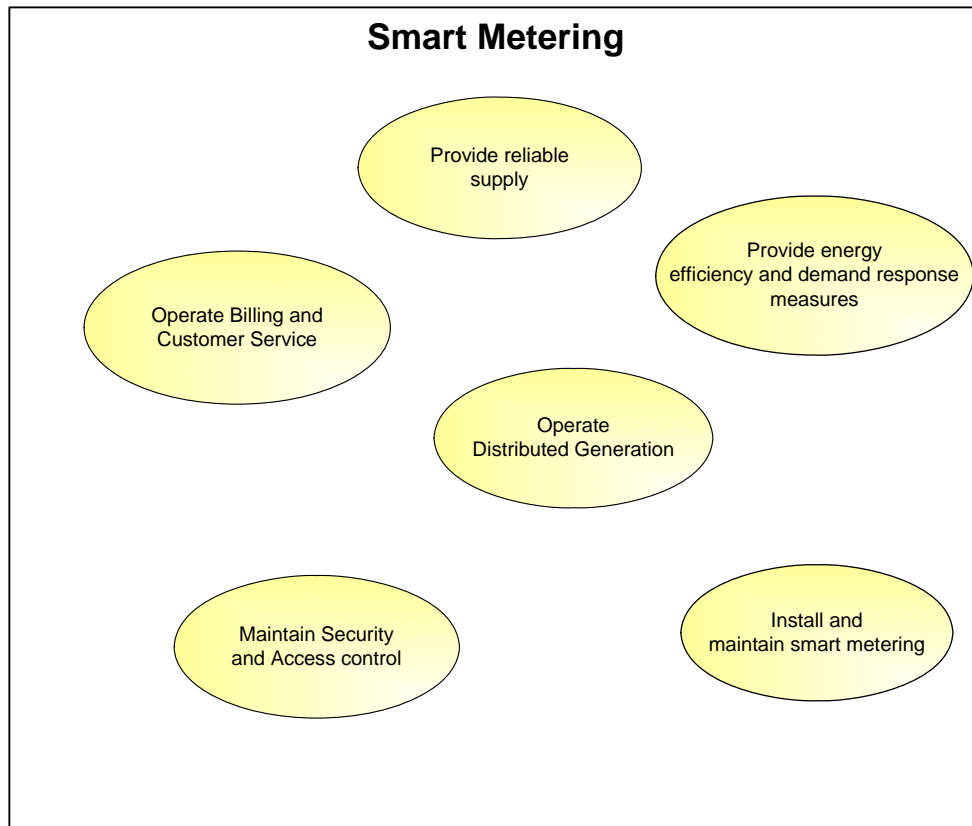
The output at each level of analysis will be use case diagrams and use case documents. The project will also maintain master lists of stakeholders, actors, assumptions, outstanding issues etc as they are uncovered.

The process at the very high level will be to:

- Identify scope of system
- Identify stakeholders
- Document the system in terms of Very High Summary Level use cases

This level very much reflects the smart metering vision.

The list might look similar to this:



Then the project will take a high level use case and drill down to a lower level to describe the component use cases.

Again identifying the actors and documenting the next level use cases.

For example: Operating Customer Service and Billing would include a number of use cases such as manage remote readings, manage tariffs, managing prepayment operations etc

Then the process is repeated to document the next level which is the functional level. For example Managing Remote Meter Readings might include set up a schedule, get an ad hoc read and so on.

This is the level where it is worth spending most effort to document in reasonable detail, as these use cases are where actors get tangible benefits.

We will compile, assumptions and issues as they are identified.

Where appropriate we will gather related information such as Business rules needed, frequency and performance requirements

Comparison with requirements of other markets

The use case approach has been used by a number of initiatives such as Southern Californian Edison, OpenHAN and the Dutch NTA specification, and at a lower level by local communications solutions. It is important to understand and document our own requirements of smart metering because they are our requirements. However there is value in comparing our use cases with functional requirements developed elsewhere and it will be essential in assessing local communications solutions.

For example, Southern California Edison has developed use cases for smart metering / smart grid. Now, their use cases are clearly not ours, as examples of differences, they are an electricity company and one that doesn't have to consider the change of supplier process. However their motivation for introducing smart metering must be similar to that of GB hence their goals should be similar. Cross checking requirements of other markets can provide a valuable suggestion of where we may have missed requirements.

Complexities and difficulties

As with any large programme there will be some genuine difficulties in documenting the requirements of smart metering. The use case methodology will be helpful in allowing progress while problematic areas are managed.

The difficulties will include:

- We don't know all the actors:
 - We don't know which organisations will be involved
 - We don't know what devices (appliances, displays, electric cars, etc) will be involvedWhere we don't know these things, it's difficult to know their requirements.
- We may not be in a position to engage with the right stakeholder
- Some uses are truly unknown currently – there are going to be requirements to “communicate with things”.

These difficulties can be managed by making and documenting assumptions or by documenting outstanding issues. The process can help identify areas where effort will be needed or other stakeholders need to contribute.

- We don't know the market model
“The system” may turn out to be many systems – for example there may be central services “systems”

For now the whole system is treated as a black box and detail of functionality of different systems can be allocated when these system are established.-

- Some uses may be commercially sensitive

These will not be documented centrally, but it would be advisable for businesses with the commercially sensitive uses to document them internally so they can test that the system will deliver those uses.

- A common problem in introducing use cases is finding conflicting cultures, unfamiliarity with use cases leading to reluctance to contribute – “we’ve always done it another way”.

The answer to this is just that it is a common problem and use cases remain a good way to capture and communicate functional requirements.

Style of Use Cases

Use cases are intended to allow easy communication of functional requirements and easy comprehension by an audience that includes the non technical. Use cases should avoid excessive length and use a straight forward language style.

There is no single template for use cases, the project will use a simple template and style guide.

The project will output use case diagrams - these will use a standard UML style.

The project will defend the style of the documentation or diagram output from over complication.

See Also

[SRSM&B.BN.09.02](#)

Options for Use Case Development

Document Information

| | |
|----------------------|------------------|
| Document Reference | SRSM&B.BN.09.07 |
| Document Status | Draft |
| Document Publication | ERA Members Only |
| Change History | 0.1 |

Appendix A: Use case Template

Use Case Name

Use case ID

Scope

Level

Description

Stakeholders

Actors

Preconditions

Post Conditions

Trigger

Basic Flow

Alternative Flows

Related Information (optional)

Notes and issues

Version History:

| Version | Date | Author | Update |
|---------|------|--------|--------|
| | | | |

Appendix B: Use Case Guidance Notes

Use Case Name

This should be in the form of a short active verb phrase, that describes the goal of the use case – 3 or 4 words should normally be plenty.

Note: you always have a goal to be successful, the title should not have the words successful or successfully in it.

Use case ID

A unique identifier for the use case

Scope

Note the system being described (in this case it will always be Smart Metering System)

Level

One of

- Programme
- Summary
- User

Note: for the higher level (programme level) use cases there is no need to document more than the name, description and stakeholders.

Description

This is a longer statement of the goal - if needed, a description of the circumstances in which it occurs.

Stakeholders

List the various entities who may not directly interact with the system but they may have an interest in the outcome of the use case. Identifying stakeholders and interests often helps in discovering hidden requirements which are not readily apparent or mentioned directly by the users during discussions

Actors

The Actors involved in the use case.

NOTE: This can distinguish the primary actor from secondary actors

Normally the actor that initiates the use case is the primary actor

Secondary actors are other parties that take part in the use case and contribute to its completion.

Preconditions

Conditions that must be achieved before the use case is started

NOTE – where a precondition exists, it usually points to the existence of another use case where the precondition is established.

NOTE – there is a “paradox” in use cases in how you handle the situation where the preconditions are not met – is this an exception? Strictly speaking you have not started the use case since the preconditions are not met and it should be handled elsewhere, but you do want to describe what happens. One approach is to rather than set any preconditions set them all validation within the use case – it is probably more convenient to describe these events in an exception flow.

Post Conditions

The conditions that have been satisfied on completion

NOTE:

You can separate the post conditions into:

- Minimal Guarantees – what is achieved on any any exit and
- Successful guarantees – what is achieved on a successful exit.

Basic Flow

The main flow of events. i.e. The steps narrating/illustrating the interaction between Actors and the System. Describe Actor’s actions/stimuli and how the system responds to those stimuli. Describe the straight and simple path where everything goes ‘right’ and enables the primary actor to accomplish his or her goal. Main flow/path should always end with a success end condition

A numbered step by step explanation of a basic successful flow.

Each step should be a short sentence that describes who does what to what.

Some strong guidelines:

- between 3 and 9 steps is a good number of steps.
Less than three and it is doubtful that a useful behaviour has happened.
More than nine suggests that some of the steps are sub functions that could warrant their own use cases.
- “validate” – don’t “check whether”

Alternative Flows

Enter any Alternative flows (extensions) here. Alternate flows are branches from the main flow to handle special conditions. They also known as extensions or Exception flows. For each extension reference note, the branching step number of the Main flow and the condition which must be true in order for this extension to be executed.

Related Information (optional)

The following information is only to be recorded where appropriate

This may be used to record:

- related use cases and the relationship:
- assumptions
- discovered requirements
- information such as the frequency of use or performance requirements.

Notes and issues

Use this section to record outstanding issues and notes that do not fit in the other sections.

Version History:

| Version | Date | Author | Update |
|--------------------|-------------------|----------------------------|-----------------------|
| The version number | The date of issue | The author of this version | The reason for update |

Appendix C: Glossary

| Term | Meaning |
|-------------------------------|---|
| Actor | A person or system that interacts with the system Note: a particular person may act as different actors at different times. |
| Level (Use Case) | Use case levels can be summary, user or sub function levels. |
| Primary actor | The primary actor of a use case is the actor that uses the system to gain benefit The primary actor is often the actor that triggers the use case. |
| Scenario | A sequence of actions that provide an example of a use case. |
| Secondary Actor | A secondary or supporting actor is an actor that contributes to the completion of a use case |
| Stakeholder | A party that has an interest in the behaviour of the system or use case |
| Stakeholder | A party that has an interest in the behaviour of the system or use case |
| System under Discussion (SuD) | The system that provides benefit to actors. It is essential that the scope is defined. The system under discussion is a special example of an actor. The system is usually treated as a 'black box' although the use case method can be used to document requirements of components of the system. |
| Use Case | A use case describes behaviour of the system that provides benefit to an actor. It can be defined as a contract between stakeholders of the system |