

# Use Case Briefs as a Rapid and Iterative Requirements Definition Tool

---

*Agile Analysis*

## **Background – Your Mission, Should You Choose to Accept It .**

..

“You have 3 weeks to define a business domain, capture the system requirements, perform a gap analysis against a package solution, and present a final report to executive management . . . any questions? [Stunned silence] No? Good. Your plane leaves 6am Monday morning. Have a nice trip . . .” [Slightly dramatised and abridged version of actual events!]

## **Overview – Use Case Briefs and Agile Analysis**

On a recent gap analysis engagement (comparing business requirements with a package solution), I had opportunity to apply Use Case Briefs (refer Cockburn 2001) as the primary tool for capturing and agreeing customer functional requirements.

Use case briefs are a minimalist form of use case, suitable for application in an Agile environment. They provide a similar level of detail as XPs “User Stories”, but with the familiarity of (in many circles) use cases.

Use case briefs answer the question “What are the goals the system under discussion must support”. They are unlikely to answer the ‘why’, and they may not tell you a lot about ‘how’ a function should be implemented. If you do them properly though, anyone who reviews them should be left in no doubt about exactly ‘what’ the system is intended to do, in terms of a clear and finite set of goals, with explanatory text, and further drill-down on detail as required. An example set of use case briefs, for the business domain, is shown at two levels below.

Actor	Goal	Level	Description
Customer	Manage Funds	0	A Customer interacts with the bank for the purpose of managing their funds, which may include depositing, withdrawing, transferring and/or checking balance of account/s.
Customer	Withdraw Funds	1	A Customer requests to withdraw a specific amount of cash from a specific account. The bank debits the account by that amount, and provides the cash to the customer. Rule: The withdrawal amount must be less than or equal to the current balance in the specified account.

## **An Agile Approach**

The exercise under discussion was required by the business to be conducted within a 3 week time box. This timeframe drove the methodology selection - we had to be able to define requirements, perform gap analysis, and document the findings, within a 3 week window. The timeframe, business scope and outcomes were fixed – the level of detail required was not.

To meet this timeframe, we divided the three weeks into 3 blocks, aligning with the abovementioned activities, ie:

Week 1: Define Requirements

Week 2: Perform Preliminary Gap Analysis

Week 3: Refine Gap Analysis and Produce Report

This discussion is concerned primarily with week 1 of the exercise, and how a complete set of system requirements may be captured in a one week timeframe.

# Use Case Briefs as a Rapid and Iterative Requirements Definition Tool

---

## *Agile Analysis*

We determined that we would apply Agile principles as much as was possible and appropriate. While it was clearly not in the scope of our exercise to deliver any actual software, we were able to adopt and commit to many of the underlying Agile principles. My post-exercise modification of the Agile principles are presented below, reflecting the principles we held to during the work. Annotations to the Agile principles: “{}” represents material I have inserted, while “[ ]” represents wording I have removed from the original Agile principle statement:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable {information} [software].
2. Welcome changing requirements, even late in {the analysis exercise} [development]. Agile processes harness change for the customer's competitive advantage.
3. Deliver {meaningful documentation} [working software] frequently, from a couple of {days} [weeks] to a couple of {weeks} [months], with a preference to the shorter timescale.
4. Business people and {analysts / architects} [developers] must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. {Meaningful documentation which enables the gap analysis exercise} [Working software] is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

## **Key Findings**

Key to the success of this exercise, were the following factors:

1. Daily 15-30 minute kick-off meetings with all key participants. These meetings ensured everyone was always aware of progress and current discoveries. There were no surprises for anyone at any time. (See principles 1,4,5,6 & 12).
2. Clear up front commitment to the agreed scope of the exercise, the process to be followed, and daily tracking against the scope and process. Daily tuning of the process to ensure scope was met.
3. Commitment by the customer to the process. (See principle 5)
4. Participation by highly skilled team members both from IBM and the business organisation. (See principle 5)
5. Regular delivery of meaningful in-progress versions of documentation (week 1 – requirements; week 2 – gap analysis; week 3 – findings report). (See principles 1, 3, 7).

## **Week 1 - Requirements Definition**

In week 1 we spent most of the time with the IBM analysis team and the customer SMEs (subject matter experts) sharing a room, utilising whiteboards and data projection, as a way to share and communicate information.

# Use Case Briefs as a Rapid and Iterative Requirements Definition Tool

---

*Agile Analysis*

Each day was treated as an iteration through that week's main objective, with completeness and depth being added on a daily basis.

## Business Scope

Day 1 was spent documenting business actors and their primary goal/s in interacting with the business. We did not want to launch into system actors and requirements, before understanding the business context, at least at a high level, even (or perhaps especially) in an exercise as compressed as this one.

The approach was text based - naming the actor, and describing them in a paragraph of text. Once all key business actors were captured. A primary goal was identified for each. Eg, if this were a banking scenario, an actor would be 'Client' and their primary goal would be 'Manage Funds'.

In this way, we were able to capture on day one, a complete, high level view of the business. This reflected one of the goals/values we stated up front for the exercise - we would provide a full breadth / variable depth gap analysis. This meant that, in the three weeks, we had committed to performing a gap analysis against the full business / system scope, at whatever level of detail could be achieved, as constrained by the 3 week time box.

On day 2, we revisited the business scope, in an attempt to take some of the 'Level 0' goals (borrowing from functional decomposition level labels), down to level 1 or 2 goals.

In the above example, this meant that the 'Client' goal of '1.0 Manage Funds' was broken down into '1.1 Deposit Funds', '1.2 Withdraw Funds', '1.3 Transfer Funds', '1.4 Perform Balance Enquiry'. Again, each of these goals was elaborated with a paragraph of text, to ensure a common understanding of each goal was maintained.

This set of business use-case briefs was completed before the close of day 2, and provided an invaluable description of the business scope, within which the system gap analysis was to be performed.

While not core to the process, a business use case model diagram and high level indicative business process flow was also documented, to provide a visual representation of the work completed.

## System Scope

Having set the scene by defining business scope, three days remained to define system scope and requirements.

We started again with actors. Not all actors in the business domain, would become actors in the system domain. In a banking scenario, an example would be a passbook-only account holder. In the business domain, they are a primary actor, as they interact directly with the business. In the system domain, they are not a primary actor - they only interact with the bank system/s via a teller. The teller would be the primary actor in the system domain for the relevant transactions.

The system domain actor list was initially compiled through revisiting the business domain actor list, to determine which business domain actors would also be users of the system. Additional primary actors for the system domain were then identified. A key set of actors identified here were the users who exist within the business boundaries. These actors were effectively invisible in the business domain, as they exist within the scope of the business boundary.

The teller interacting with the passbook-only customer serves as an example here. In the business context, the passbook-only customer interacts with the business, with a goal to Manage Funds. How the business meets the customer's goal is invisible in the business context.

# Use Case Briefs as a Rapid and Iterative Requirements Definition Tool

---

## *Agile Analysis*

In the system context, the teller becomes the primary actor for this set of transactions, and the passbook-only customer is effectively considered as out-of-scope. (There are various approaches to still capturing this customer's interests in the system, but these will not be elaborated here).

Having established the list of system actors, with description, the elaboration process was repeated - defining level 0 goals, and then iteratively and incrementally building detail as required into level 1 and 2 goals / use case briefs.

### **Sidebar - What not How**

A problem which often occurs in defining business requirements, is capturing too much of the as-is, or 'How' a function is carried out, rather than understanding 'What' the function is, and 'What' it is trying to achieve.

This is never more apparent than when performing a gap analysis between business requirements and a package solution – and this is where use case briefs and Agile requirements management are so valuable.

Take the example of the Client who has a goal to Withdraw Cash from an ATM. If we look at this Goal, and compare with a package banking product, we should expect to find a high level of fit / low level of gap, between the business requirement and the package solution (if the package banking solution does not offer this feature, we may need to reconsider its inclusion in the gap analysis).

If, however, we look at the as-is implementation by the incumbent solution, or perhaps even capture too much detail on how the business imagines it could or should be implemented, we are likely to encounter a much lower level of fit, and many more gaps. The issue here is one of confusing a business requirement with the systems solution to that requirement. The more 'How' (design) elements the business defines into the requirements, the more they constrain the solution – possibly losing opportunity to innovate, adopt best practices, and minimise costs associated with customisation of the package solution.

The use case brief / Goal approach to requirements specification assists through keeping requirements at a goal + explanation level (the 'What' a solution must do), rather than a solution level (the 'How' a solution meets the goals).

### **Conclusion**

By the close of week 1, the requirements document was completed, defining business scope, system scope and functional requirements.

In parallel with this activity, which was focussed on the functional aspects of the problem, a data entity model was constructed to define the static aspects of the problem domain. Non-functional and architectural issues were captured (eg transaction volumes, legacy system impacts on solution).

Without detailing the subsequent steps (perhaps in a future article), the remainder of the exercise was completed within the required time frame, and the exercise was regarded as a success. What's more, the customer expressed a keen interest in the methodology used, and its applicability for subsequent and larger stages of the implementation lifecycle.

While there is certainly an argument that use case briefs are not suited to all engagements, or perhaps even all phases of a single engagement, they are a useful tool in an Agile environment, and one which I look forward to using again.