

Writing Better Requirements

Jan Kusiak

IRM Training Pty Ltd ACN 007 219 589

Suite 209, 620 St Kilda Rd, Melbourne, Vic. 3004, Australia

Tel: +613 9533 2300

training@irm.com.au www.irm.com.au

Overview

There's little argument that investigating and identifying business needs (i.e. requirements) is a critical task of business analysis. However it's of little use correctly identifying business needs if we can't then effectively document them - to the clients who will be paying for the solution and to the developers who will be building it. In today's time poor world we need to address both audiences in a single document.

This paper – based on IRM's *Writing Better Requirements* workshop – takes a top down view of the requirements life-cycle. The paper looks at where business requirements come from and what analysts can do to turn them into solution requirements that developers can work with. Understanding the end-to-end process is a first step in producing well written, clear and specific requirements documentation.

The source of requirements

As we know, every organisation has (or should have!) a business strategy which describes how it will achieve its objectives, its goals. For business units within the organisation to achieve these objectives, opportunities need to be seized or problems overcome.

Equally, every organisation has a framework in which it operates. The framework ranges across many areas - customer needs, market forces, regulatory, how it's organised and structured, etc. Of particular relevance to the business analyst is the enterprise architecture, the blueprint of the organisations processes and systems.

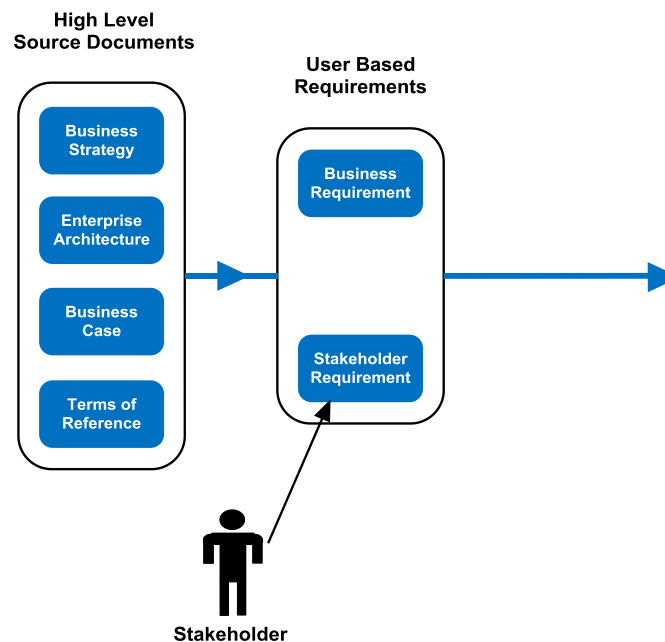
The business strategy, together with the enterprise architecture, give business units a framework to achieve their business objectives.

To seize opportunities, to solve problems, we need to make things happen and this is done through projects. Without wishing to upset anyone who might favour one project methodology over another, it's fair to say that in some shape or form every project has a business case and a terms of reference. These may be written down or in someone's head, they may be set in stone or fluid and changing. Either way, the business case should be telling us what we're trying to achieve and the terms of reference should tell us (or at least give us a framework) of how it will be done.

What we're trying to achieve – be it removing a supply chain roadblock or launching a new product – is the essence of business requirements.

The terms of reference together with the business case, enterprise architecture and organisational strategy, give us our high level source documents for the project.

One of the golden rules of business analysis is to keep an open mind. As such new information (e.g. extra functionality, performance constraints) is often identified by stakeholders during the investigation and analysis phase. If approved, these additional requirements need to be added to the overall scope of the project.



The high level source documents drive business requirements and these business requirements, together with stakeholder requirements, can collectively be called user based requirements. Each one is documented in a template and forms part of our overall single document along with references to the high level source documents.

Although stakeholder requirements translate into business and/or solution requirements, it's important not to lose track of their individual needs. The needs of stakeholders should be easily identifiable, after all they hold a stake in the project and can make or break it.

From business requirements to solution requirements

So how do we turn the general into the specific – how does a new business function become a solution requirement? First, let's define what we mean by a solution requirement as it consists of two components:

- Functional requirement: describes a business capability e.g. process a customer enquiry
- Non-functional requirement: describes an environmental condition e.g. 1 second response time

Specifying non-functional requirements is usually straightforward. A 1 second response is a very clear statement of what's required although on its own this is not enough. We may also need to specify for how many concurrent users, for what type of enquiry, etc. However a template driven method (fill out the boxes) helps to cover most variables.

Functional requirements need a different approach so rather than describing them using just words and sentences we can use a technique called event analysis.

Event analysis is a process whereby each functional requirement is defined as an activity, action or process that needs a response. The resulting event is then broken down into a number of components e.g. input, trigger, process function, output, etc. Multiple events (functional requirements) can be then be listed in a table with a consistent format for each.

Event name	Trigger	Initiator	Process or Use Case	Data Store	Output	Recipient
Product purchase	Online order	Customer	Process order	OrdersDB (W)	Order confirmation	Customer & supplier
Order status enquiry	Online enquiry	Customer	Check order status	OrdersDB (R)	Order status	Customer
Customer statement	End of month	System clock	Produce statement	AccountDB (R)	Printed statement	Customer

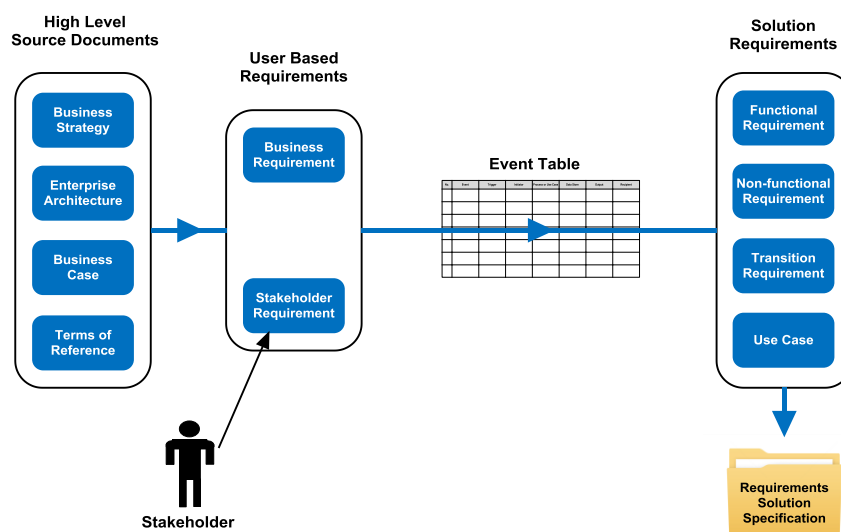
Events are not rigid specifications, their components vary depending on how and where they are used. In the following example we've included a parameter for the data store (the database name) but in a pure object-oriented development project, this parameter might be omitted.

Event analysis bridges the gap between business requirements and functional requirements. Once we've got a clear and consistent picture of each business function we can further specify it using any one of a number of modelling notations – UML, DFD, BPMN, etc. For more on event analysis see the IRM article [Event Based Analysis & Modelling](#).

A point on diagrams and modelling: sometimes a diagram is not enough to describe precisely what occurs in a business (functional) requirement. An activity or BPMN diagram can show us the flow of an event but can become unmanageable when trying to accommodate alternatives e.g. give the customer 3 attempts at entering their PIN before locking them out. We need to use words and sentences to spell out each step of the event but rather than using free-form text, a common technique for doing this is the formal use case. For an example see [How to use Use Cases](#).

Write once, read many

Traditional document deliverables (a *Business Requirements Specification* for the users and a *Functional Specification* for the developers) are fast giving way to a single document for both audiences. Few people have time to read (let alone write) two documents about the same project. The challenge when writing a single document for users and for developers does however draw very much on tradition.



To make a single document approach work, there must be a clear and identifiable difference in the way that each type of requirement is documented. Business requirements summarise the what and why of a project, solution requirements explain this in detail.

Oh, and don't forget any temporary requirements needed during the migration from As-Is to To-Be. (the transition requirements). These requirements might be identified anywhere during the analysis phase but cannot be finally nailed down until the functionality of the future system (or the next release) is signed off.

Forward and backward traceability

It's stating the obvious but every document and requirement should have its own unique identifier. The flow described in the preceding diagram:

Source Documents >> User Requirements >> Solution Requirements

will give us forward traceability. We can then show how business strategy (or the business case) drives business requirements which in turn will be delivered with solution requirements. Similarly, when asked *why are we coding this?* we can reference it back to a specific business function and in turn the project's source documents.

Templates for documenting requirements

Every organisation has its own unique needs and will adapt off-the-shelf templates to suit their purposes. However all templates have many common characteristics and we've combined these into the template used in the *Writing Better Requirements* course.

Similarly we've taken the most common topics from off-the-shelf *Business Requirements Specifications* and *Functional Specifications* and combined them into a single [Requirements Solution Specification](#) template. The template is not copyrighted and can be modified to meet your own needs. Follow the link if you'd like a copy.

© IRM Training Pty Ltd. All rights reserved.

Send feedback and comments to: training@irm.com.au

You may use this article free of charge provided you do not alter it in any way and include all copyright notices.