

Just Enough Documentation

Jan Kusiak, Training Services Manager
IRM Training Pty Ltd ACN 007 219 589
Suite 209, 620 St Kilda Rd, Melbourne, Vic. 3004, Australia
Tel: +613 9533 2300
www.irm.com.au

Overview

Is documentation a blessing or a curse? If you're working on an agile project does it get in the way? If you're updating a core system that runs your company's business, are you cursing the analyst who didn't adequately document all the business functionality? Is today's agile project tomorrow's core system?

How much documentation to produce is one of the most troublesome issues facing analysts today. There are no hard and fast rules on this and successful projects define their own rules to fit the circumstances and organisation.

However one thing every project has in common - there is always some documentation even if it's only a terms of reference. Not many projects get signed off based on a verbal business case - at least the company accountant hopes not! Here are some guidelines to consider when producing your next set of documentation.

Follow the money – the project Terms of Reference

Everything should start with a user (client) request (also called a Terms of Reference, Project Charter or Project Brief). The "user" can be a business unit or the users of a system – it can even be the IT Department if it's an infrastructure project. The important thing is that it comes from the people who will fund the project.

The Terms of Reference (TOR) is a high level business document which states the business need or problem. It's not a solution or design document and it's certainly not a requirements specification.

A TOR is a two or three page document which can be shown to any manager or executive so they can immediately grasp the objectives of the project and what problem or opportunity it is addressing. A TOR should also identify the scope and constraints of the project plus give some indication of the amount of money the client is prepared to commit.

It's a wise idea to avoid making financial estimates sound anything like a business case. How often have you been caught out by someone saying *".....but you said 6 months ago the project would only cost \$150,000. Now you want more?"* Far better to say something like *"....the client is prepared to fund a 6 month work plan up to \$150,000 and then review project deliverables before committing to further funding."*

Most forward thinking organisations get the business analyst involved at the TOR stage. Some organisations on the other hand ask a project manager to write the TOR. This can get people confused as they think it's a project manager's job. It's not, the PM is doing a BA task.

Does the client know what they want?

This is not a trick question. They will usually know exactly what their problems are or what business opportunities they're chasing. But do they know the best way of tackling them?

Business analysts will often get involved in a feasibility study as a precursor to (or as part of) the TOR. This is where you add value by performing the "analyst" component of your job title. You'll be analysing current business processes and modelling potential future business processes.

Again, a number of people will want to understand your analysis work and a feasibility report, paper or online, serves this purpose very well. The report explains exactly what the problem or opportunity is (and isn't). The report also identifies one or more ways of addressing them.

Describing functionality – user stories or requirements specifications?

Once your client has agreed the contents of the TOR - that is, what problem or opportunity they're addressing - we can start to itemise the changes to the business. These can be modifications of existing processes or totally new activities. Processes and activities are made up of functions (what the system does) and it's the documentation of these functions, called requirements, which causes much debate. Should we document all the requirements first – should we document them as we go – should we document them at all?

Questions like these can cause ongoing problems for organisations. Business analysts can spend so much time identifying and documenting all the business requirements that the document becomes so big it's unusable. Worse still, business priorities may have changed and the document is worthless.

Traditional software development methods have caused many of these problems but it's not because the methods themselves are wrong. Rather it's that the IT community hasn't moved with the times. Not that long ago a three year billing redevelopment project at a major service provider involved a project team of 250+ staff. The wages bill alone was \$25m per year. This organisation wanted the functionality of the new system to be documented and signed off in advance, they couldn't afford to have teams sitting idle without a detailed work plan.

The same is true of regulatory and legislative changes required of banks and government departments. These organisations are obligated to document all the changes needed, then plan a rigorous schedule to ensure that they're implemented by the due date.

The bigger the project, both in terms of cost and importance to the organisation, the more they lend themselves to a formal step-by-step methodology such as waterfall. A comprehensive specification, usually called a Business Requirements Specification (BRS) or Business Requirements Document (BRD), is almost mandatory. With the complex functionality of today's system a BRS/BRD can often run to several hundred pages or more.

But aren't small projects important as well? Yes they can be and they may demand a formal, structured documentation set just like a core system. The real issue is for documentation not to hinder or get in the way of the success of the project. If you're involved in an iterative or agile

project with requirements being identified or prioritised as the project progresses then by default the BRS/BRD will be an evolving document that gets progressively fleshed out.

So what format should the BRS/BRD take? Dozens of templates exist, many are freely available on the internet and most of them take something from the IEEE standard *Recommended Practice for Software Requirements Specifications* (IEEE Std 830-1998).

This is a serious document which will serve you well. We all tend to like templates because they give us structure and because they've been used before so they must be ok.

However there's nothing to stop you using your own format for a BRS. If you're using UML diagrams on your project then you can document each business requirement with a use case diagram and a textual use case description. Similarly if you're using a story card for each requirement (e.g. on an agile project) your BRS can look like a box of index cards.

Whatever the type of project though, the BRS/BRD remains something that explains what the clients problems are and what's going to be done about it – but it must explain this in terms the client can understand. Keep it simple but no simpler.

Talking to the technical folk

With all the focus on business requirements and client needs it's easy to forget that something still has to get built and the people who do this are software developers – either your company's or an outside supplier. The business analyst always has two roles – communicating with clients and communicating with developers. When it comes to turning business requirements into code the developer will appreciate a clear description of what you want.

Traditionally this was done by producing a Functional Specification, a logical description of the business requirements. This would comprise process flow models, business rules, data models, data definitions, event models, etc., enough information for architects to start the software design process.

Today we still need to give architects and developers sufficient information to do this. For example on a project using UML notation, we might use a Use Case diagram and an informal (or essential) Use Case to provide a plain language description of the business functionality needed. The diagram and essential Use Case would, in most circumstances, be sufficient for a Business Requirements Specification and for communicating with clients. For communications with developers we could add a formal Use Case which describes every primary and alternate flow of events – that is all the business rules of the business function. (See the paper [How to use Use Cases](#) for an example of this). If the business function is particularly complex we might also add an Activity Diagram to describe what is happening inside the Use Case.

On agile projects with a focus on timely delivery of working software, documenting business functions may not be a critical issue. After all if developers and clients are sitting next to each other then why write elaborate descriptions – just lean over and ask them! However never forget the issue of ongoing application maintenance (see the following section).

Agile methodologies have gained a high profile through the principle of developing software as a primary goal (and minimising unnecessary documentation) but it's worth keeping in mind agile's

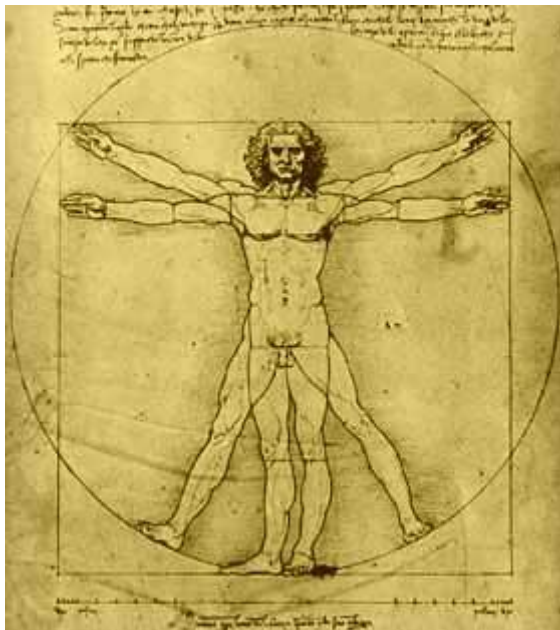
secondary goal – enable the next effort. This means not only re-usable code but ease of maintenance and future enhancement. To know where you're going in the future you usually need to know where you've been.

This paper hasn't dwelt too much on requirements management systems but for large scale and/or complex systems this can be the only way of tracking interdependencies between business functions and providing traceability between a requirement (business function) and working code.

Modelling

A word on modelling which often get forgotten as a fundamental business analysis skill. Put simply modelling means drawing pictures – but using diagrams which have rules and conventions.

Engineers and software developers like pictures. Not because they can't read but because they are wary of the ambiguity of the English language. There can be so many different interpretations of a single sentence. On the other hand, diagrams that follow rules – such as UML, BPMN, ERD, DFD – work best because they leave minimal room for misinterpretation.



An attempt to describe mankind in all his proportions, the form of limbs and all functions in detail, solely with words must be discouraged. The reader will become increasingly more confused as the writer describes more and more detail. It is absolutely essential to draw as well as to write.

Leonardo de Vinci

Something else to remember about modelling – it's also a great tool for analysing a client's needs or problems. The act of drawing forces us to take a systematic look at what is being investigated.

Much like taking notes in a meeting helps us remember, so drawing pictures helps us think.

Maintenance – it comes with the territory

What's the life of an application? You only have to look at some of the core systems running big business to find applications which are 10, even 20 years old. For example the database

management system Model 204 (first released in 1972) has been running Australia's social security and welfare payments system since 1983. It's still going strong 27 years later.

There's no such thing as a throwaway system (unless it's a prototype) and it doesn't matter what the development methodology (agile, waterfall, iterative, etc), all systems are expected to have a productive life. This might be 6 months or 6 years but one thing you should never rely on is that the original project team will stick around. Just like disaster recovery planning, any new business analyst should be able to come along and pick up where the previous one left off. It should be easy for them to understand the current business functionality of the system. If they can't easily find out what it's meant to be doing they will curse you!

So how much is enough?

One school of thought is to only produce documentation if the stakeholders ask for it or are prepared to pay for it. On the surface this is reasonable, after all to first produce documentation and then to set up a system to keep it relevant, takes time and effort.

However not all stakeholders have the best interest of the organisation at heart. How often have you seen someone chase a quick win to get that promotion or high-profile transfer. Keeping project costs low achieves this but at the expense of potential problems down the track.

For most analysts the reality is they will produce as much (or as little) documentation as their manager tells them to or as much as company standards dictate. We often can't do a lot about this one way or the other. One thing we can do however is to make sure any documentation we produce serves its primary purpose which is to communicate with the reader – documentation should be written with the reader in mind.

Management wants to know what the project will achieve - what business problem it's solving - what opportunity it's exploiting. A Terms of Reference or Project Charter will answer this question.

Business users are primarily concerned with what the system will do and an index card (user story) approach or catalogue of Use Cases can do this quite effectively while still recording each business function.

Communication with architects (solution designers) and developers (solution builders) is the final piece in the documentation jigsaw. Whilst most software development tools are self-documenting and will tell you how the software works, they won't tell you what the business is using it for. This is the business analyst's job and, as mentioned before, the tools of the trade are diagrams and models.

To put the previously mentioned Leonardo de Vinci quote into a modern context – if you want to save lots of re-work, use a picture.

© 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.