

Enterprise Learning

The Beauty of Requirements

Requirements form an integral part of the design process

Enterprise learning is a term that describes the coordination and implementation of learning initiatives across the whole organisation. When implemented correctly and integrated into the enterprise this has been shown to lead to learning and development cost efficiencies, consistency in communications and improved productivity. Enterprise learning also improves outcomes by using every learning event as an opportunity to promote the values of the organisation, to integrate cultural change and to consider the whole-of-business objectives. And this, I consider, is a good thing.

However enterprise learning usually has one major challenge to overcome before it can ever be successful: its own ambition. That is it usually involves the whole enterprise, whether large or small, either directly or indirectly. And as we have all experienced when too many vested interests get involved, large initiatives such as enterprise learning programs can get bogged down in their own detail.

Virtually all enterprise-wide initiatives follow a simple iterative plan of design, develop and deliver. Whether the processes remain as manual processes or get codified into a software application they still require a plan, or a set of blueprints, that describe the components, actors and interactions in the process. These are typically driven by organisational opportunity or need and guided by business strategy and objectives, purpose and technology constraints and available resources.

Where do we start to manage all those interests, expectations and precious budgets? I find it best to start by gathering requirements. While managers and developers understand that good requirements help describe the project and outcomes many are unsure of exactly how to gather and use them. This article will explain how and why the methods employed by software developers and large systems analysts can lead to clearer outcomes and reduced project risk. And while the concept of requirements came from the software world they are

now utilised in any organisational project that requires good planning and consistent outcomes.

What are requirements exactly?

Every requirement states a need and reason, or a description and rationale. For instance when sitting down with an architect for a discussion on designing and building a new house we would state our need for a number of functional requirements like so:

- We need three bedrooms and two bathrooms
- We will require a two car garage
- We need a large living room
- The kitchen must cater for two cooks

However we may also want to stipulate a number of requirements that are non-functional yet still necessary and these will need further negotiation to clarify their meaning.

- We want lots of air and light
- We want a relaxing atmosphere
- We want a contemporary design
- We don't want 'heavy' building materials

Requirements fall into two major categories – functional and non-functional. They form the agreement, the contract, between you and the developer or vendor. They are not the design. They are the

Why and the what, the When and the Where but they are not the How! In the above case it is the architect's job to deliver a design for

the house based upon your functional needs, your non-functional desires and constraints (including budget) and all with an aesthetic pleasing to you the client. They then work closely with the builder to implement their design in a timely and cost-effective manner.

We have been building houses for several millennia. We have only been designing commercial organisational processes for a couple of centuries and only codifying

those into software applications for around 40 years. Software application programming is often more art than science as we try to capture abstract ideas in a concrete yet inexact form (software code). We are slowly getting better at it, and requirements are one method to help ensure the initiative we spend time and effort on developing actually delivers.

While the functional requirements have been the domain of the developers and programmers research has shown it is the non-functional requirements that dominate the reasons for failure or success in most new programs. Functional requirements are the hard, objective features and processes that any program, whether enterprise learning or a new software application, must deliver to meet our desire or needs, usually defined by a business/organisational opportunity or threat. They are relatively easy to define and describe.

Non-functional requirements on the other hand are 'soft', subjective and difficult to describe. They are the properties or qualities a finished product must have. Suzanne and James Robertson, developers of the Volere Requirements Specification Template, list 27 major requirement types and in addition to the functional and non-functional categories also include project drivers, constraints and issues. The non-functional requirements outweigh the functional by 24 to 3.

It is easy to see why the non-functional requirements are so important to the success of the program. They include all the contentious areas such as Look and Feel, Usability and Performance. All of these areas are capable of dividing opinion and support for any project. They are described by such terms as good and bad. And we all agree on what good and bad mean don't we?

There is often a long way between the promise and the product and you will have to pay for the journey.

Requirements are what you and I, the end users, expect of an application, program or process. However, requirements are getting a wider audience than just the software application developers. They are being used more and more in the overall project management of new organisational initiatives. Look at a new enterprise learning program as an example. We know it will have not only business objectives but also learning and development outcomes which also have preferred methods and existing constraints. It also needs to ensure the organisational culture is either revamped or retained, and will have certain regulatory and market place requirements to ensure its viability and smooth implementation. And all of this is before we have even discussed the appropriate technology! So the first phase of any new program implementation would begin with the requirements gathering of these mainly non-functional, but very necessary, requirements.

Let's now consider that some of your business requirements suggest an online or blended learning approach would best suit your needs. We would need to begin the application and vendor selection process to find a solution. Now this is where our more stringent functional requirements are required, and these are usually non-negotiable. You would certainly be less than impressed if your architect came up with only one bedroom and a bathroom in the garage. Yet this is the equivalent of what happens in the software application world all the time. And only a relatively experienced programmer will ever spot the obvious flaw in the design. They will then tell you they can customise this feature for you. Whenever you hear the word customise just think money. There is often a long way between the promise and the product and you will have to pay for the journey.

In a house when you buy a refrigerator

you can specify which way the door swings, left or right. (This is a requirement.) This is so the door doesn't swing open blocking access to kitchen cabinets or stifling workflow while preparing food. Yet the metaphor of the refrigerator door swinging the wrong way is with us all the time in software. And we just put up with it! There are usually decisions made by the developer that don't suit our workflow or situation. We put up with these because the cost of fixing the problems is prohibitive. Vendors are getting better at designing flexible modular applications which are more easily customised, yet even these are designed for the 'typical' market needs. Try to go outside the square and you'll be on your own.

So your requirements not only work as part of the selection process but they also form an integral part of your design process. The design is always what is left after needs are satisfied and all constraints are met with (hopefully) a dusting of innovation. Unfortunately design involves many compromises, but without the guiding light of requirements it becomes almost impossible to define a product you will be able to develop, use, and integrate into existing and future systems in a cost and time-effective manner. What you don't want is your software application to pose unnecessary burdens on end-users through unnecessary compromise. The application may be rejected outright (or even worse ignored) and implementation will fail.

Once again requirements can be your saviour. The objective is to have all your requirements gathered and understood at the start of the design/selection process so the task to find the application/vendor with the closest fit to your needs is made easier. If you have little, or scant, knowledge of your requirements and can't articulate them then you may have to change your business practices, methods of purpose and even technology platform

to suit the vendor. This is akin to the tail wagging the dog. Of course commercial off the shelf applications may suit you if your needs are modest or you don't have the skilled resources to customise and so on. However if you don't know your requirements then finding out your shortcomings after the event may be a costly and continual annoyance.

To finish on a bright note though there is one final reason for developing good requirements: making people happy. Good requirements should replace confusion with clarity, conflict with agreement and suspicion with trust. All of these form better relationships between stakeholders in any

enterprise wide project, including external vendors and associates. Just as a clear blueprint drives any construction project good requirements form a conduit for clear communication and successful and cost effective outcomes. ☒

Next month I will take you through the requirements gathering process using such 'tools' as client interviews, developing Use-cases and prototyping.

Robertson, J, Robertson, S, 2005, "Requirements-Led Project Management", Addison-Wesley, Boston, USA.

Paul McKey is Managing Director of Redbean Learning Solutions and writes and speaks extensively on the subject of online learning. He incorporates a background in business, technology and education to bring a unique, creative, and commercial perspective to the implementation of learning in organisations.

Redbean Learning Solutions is an international learning and development organisation specialising in the design, development and implementation of online learning programs. Redbean can provide better business outcomes for your organisation through an independent and objective approach to helping improve your people, product and practices.