

1. Getting requirements right the first time (Part 1)

1.1. Introduction

You know how it goes: 'If I had a dollar for every time...'; well, this time it's about requirements. If I had a dollar for every time I hear business people, developers and business analysts say 'it's impossible to know all your requirements up front'.

This article is the first in a series that expands on the Myths of IT development article I wrote last year. In that article, Myth 2 was about it being OK that requirements change throughout the lifecycle. This myth is perpetuated by the mistaken belief that it is impossible to know all your requirements up front, or that the requirements will dramatically change in 6 months.

This idea also tackled, but from a different angle in Myth 9, where I showed that because developers believe it's impossible to know the end goal that they should build a bit and use a bit, developing the application incrementally.

In this article, I want to make the point that in most IT projects, the requirements do change 'all the time', but not for the reasons that people think. The main reason that requirements change throughout and after the lifecycle is that are not captured correctly in the first place. It is precisely because they requirements were not right to start with that they need to change.

Let me repeat the point: it is not because the requirements fundamentally change. I'll show you that if you capture the right requirements in the right way, you can stabilise your application and avoid the hundreds of change requests every quarter.

It is a truism that if you don't know where you're going, then any road will take you there. I'll show you how to get those requirements right the first time. Let's start with why requirements, as they are currently captured, are inevitably wrong and will need to be changed regularly.

1.2. Why requirements notoriously difficult to pin down

1.2.1. How requirements are generally captured

When I ask people how they capture requirements and build applications, the answer is usually one of the following:

- ◀ The developers are told to build something, they do so, then show it to users and begin iterating the application because it's not right.
- ◀ Developers survey or interview users on what they want, then they go back and build something, then show it to users and begin iterating the application because it's not right.
- ◀ The BAs run workshops with end users, asking them how they do their work and what they need, model it, hand it to developers, who build something, then show it to users and begin iterating the application because it's not right.
- ◀ Business process consultants figure out the best way that users should do the work, tell the developers, build something, then show it to users and begin iterating the application because it's not right.

Do you notice anything similar between the responses? In all cases, iteration is inevitable because what they produce is not right. Why? I've found the following tends to happen:

- ◀ Users who are skilled at a certain job cannot articulate how they do it, because it has become an automatic skill.
- ◀ Users do not really know what they want, they just do what they're asked to do. While they can identify areas for improvement, they're usually superficial.
- ◀ Most people doing a job learned it from their colleagues or boss, who learned it from their colleagues and bosses, and so on, until what they are doing now bears little resemblance to the long ago initial instruction.
- ◀ In workshops, users are removed from their work environment and therefore need to remember what they do. They don't recall the detail or the automated skills and tend to be swayed by group think.
- ◀ Most interviewers are not good at interviewing, and get stuck on the superficial differences between people and roles, without seeing the fundamental commonalities.
- ◀ Most developers and BAs do not know how to analyse work and people in order to identify what contributes to high performance.
- ◀ Most users have a limited understanding of 'best practice' and the broader domain or discipline they're working in as a result of being specialised or doing things the way the organisation prefers to do it. Therefore, asking them what they want will usually deliver more of the same.
- ◀ Most developers and BAs have little understanding of the domain of the application they are developing, and so make gross assumptions or miss the critical components.
- ◀ Many developers look at other similar programs, identify the key features and replicate them in the new application, using the former as a source of requirements.

I remember one time I was conducting an onsite observation of a person working the finance department performing a number of financial transactions, such as paying bills and salaries. She completed the transaction using the current version of the application. I asked her 'how did you find doing that transaction?' She said 'fine.' I then said 'let's go through it again and I want to ask you about a specific aspect of the process you went through.' We went through it again and I stopped her at a certain point and said 'did you notice that you repeated those steps in the process three times, one after the other?' She looked at me, amazed, almost incredulous and said 'I've never noticed that before'. She then cancelled the transaction so she could start again just to check if it happened each time she did it.

What's the point of the story? People become so automated in their use of applications over time, usually because they are so poor, that they simply remember the sequence of activities to perform without thinking about them. If you simply ask people what they want, or how the application should work, they'll usually say little or speak in terms of what they know or are aware of. It is unlikely that people deeply analyse what they do in order to determine more efficient and effective means.

From talking to developers and BAs either in development firms, in client organisations or when I have job interviews with them, there is a consistent set of assumptions they hold:

- ◀ Every organisation is vastly different
- ◀ The same jobs within and across different organisations are vastly different
- ◀ Asking people what they want is a sufficient way to capture requirements, in interviews or workshops
- ◀ Looking at other applications is also a good way to capture (functional) requirements,
- ◀ Developers and BAs know enough to fill in the gaps and work out what to build
- ◀ The business knows exactly what it wants
- ◀ The business has no idea what it wants
- ◀ Iteration is the only way to go because you can never know all your requirements in advance

And the most dangerous assumption is:

- ◀ Since the requirements will change all the time and if I get the application wrong, which will probably be the case, then it's not my fault because I was building what you said you wanted.

You might think I'm being a bit harsh, but just take a look at the quality and usability of most applications you use.

Rather than looking deeper into why these assumptions are held, let's look at what to differently when capturing requirements.

1.3. Challenging the assumptions behind requirements gathering

In another article, the Psychologist as business analyst, I talked generally about requirements and the role of the industrial / organisational (I/O) psychologist in capturing requirements. I outlined how an I/O psychologist is able to evaluate people's performance, analyse jobs, redesign jobs and improve people's performance.

If you accept the view that improving an application is an exercise in improving people's performance, then you must know how to improve performance in order to design an application that will actually do that.

Let's talk about three of the assumptions in some detail:

- ◀ The similarity of functions and jobs across organisations
- ◀ Asking people what they want is sufficient
- ◀ Reviewing other applications is a good source of functional requirements

1.3.1. How similar are specific functions and jobs across organisations?

In a recent meeting with a developer, I asked the question 'do you think the way people perform sales in one bank is the same as people in another bank?' And without taking a moment to think about it, he said 'of course they're different'.

What do you think? Are they similar or different?

If you take into consideration that each organisation has some similar and some different products, that the people are different and so have different styles, and that the organisations have different cultures and management, then you would think that the two jobs, even though they have the same title, would be different.

But are they really that different? Let's answer the question by examining the fundamental activities within the sales job. The typical sales process is something like this:

- ◀ Identify a pool of prospects
- ◀ Make contact with the prospects to set up a meeting
- ◀ Identify needs and present the features and benefits of appropriate products
- ◀ Handle objections
- ◀ Close the sale
- ◀ Get the money
- ◀ Provide ongoing service
- ◀ Attempt to cross and up-sell other products

In general, the nature of people's work does not dramatically change over time. In the sales example, above, same basic processes have been used for thousand years. It doesn't matter what you sell or what your target market is like, you need to follow the same basic process. While the core process does not change, the superficial features do change:

- ◀ The products and services being sold,
- ◀ The particular markets being sold to,
- ◀ How to present features and benefits,
- ◀ How to handle objections,
- ◀ How to close the sale.

But none of these features requires or mandates a change in the process. Therefore, if you start with the fundamental process first, you are more likely to get the requirements right the first time. You can use the fundamental process as a backbone for your application whereby the process steps stand the test of time, but the specific detail within each does not need to. So when it comes to make a change, the backbone stays in place, keeping a familiar structure and framework for people to work with.

The similarity of like-jobs across organisations also extends to the similarity of organisations themselves. It's generally accepted in change management and organisational development circles that most organisations are mostly the same.

I imagine the first thing you're thinking is 'no way, my organisation is completely different to anyone else!!' but take a moment and think about it. What are the main functions in your organisation? I'll bet it contains many of the following:

- ◀ Human resources
- ◀ Finance / Accounting (including accounts payable and receivable departments)
- ◀ Marketing
- ◀ Sales
- ◀ Operations
- ◀ Logistics

Do you think that your finance department is really different to another organisation's? If we all have to meet certain standards regarding financial reporting, then there aren't that many different ways you can do things to meet the same end goals.

Just like with the sales example, earlier, it is certainly true that different organisations make and / or sell different products and services, but the underlying fundamentals are very similar.

If jobs and organisations were really so different and so specialised, you would never see people moving jobs because the learning curve would be simply too steep.

1.3.2. Asking people what they want is sufficient

Most developers think that if you want to find out what the user requirements are, then you need to ask them. The assumption is that because they are performing in the job, they can tell you what they need and how their job can be improved.

Consider these issues:

- ◀ If you asked your team how they do their job, would you get similar or different answers from everyone?
- ◀ Does everyone in your organisation operate at a high level of performance? How do you know you're asking the right people? What happens if you ask a low performing person what they want, its then implemented, and everyone then performs at the same low level?
- ◀ If people do tell you what they want, how do you know it's right? How do you know it's what the organisation really needs to improve performance?
- ◀ If people tell you that they need function X and Y and Z, how will you actually deliver it? There are hundreds, if not thousands, of different ways to implement a certain function. Which way do you choose?

If you're going through an application redesign at the moment, based on user research like this and you're feeling a little uneasy at the moment, then I'm not surprised. It is a fact that developers and BAs are not taught the critical skills to research, analyse and diagnose human performance.

The other thing that people don't know is that as you become expert at something, and therefore presumably a good source of information, the less you can describe what you are doing. I'll dip a bit into psychology and the organisation of memory for a moment to cover this issue.

As people learn how to do a job, they develop knowledge structures about the domain. This is a mental model of how to do the job. If they've been exposed to a structured training program that is clearly mapped to the formal domain (e.g. accounting), then they will likely develop a quality mental model of their work. If they've learnt by on-job-training, then it will likely be very specific in nature, incomplete and possibly wrong. Their knowledge has been handed down from one person to the next, reinterpreted and changed. You know how 'Chinese whispers' works. you can therefore bet that people apparently doing the same job do it differently. No wonder BAs and developers have a hard time making sense of all the differences.

Further, as you become more expert, the knowledge and skills become more and more ingrained and automated. It becomes harder to change and it is harder for people to articulate what they do. Most times when we ask an expert how they do things you get back something like 'you just do it'.

1.3.3. Reviewing other applications is a good source of functional requirements

The third assumption we'll address is using other similar applications as a source of functional requirements. Sometimes developers will use the older version, or look at competitive applications for ideas on what to include.

If you accept that it's easy to focus on the superficial differences between jobs and organisations, and that asking users is not always the best source of requirements when building applications, and you agree that most applications are built on these assumptions, then what do you think the quality of those other applications is likely to be? Do you really want your developers and BAs using them as source information on how to do things?

In just about all complex applications that we review, we find the following:

- ◀ Training users in using application is about teaching the specific sequences of buttons to click to achieve some goal.
- ◀ Training is never about understanding the domain such that if you teach people how to sell, then they can use the customer management application.
- ◀ Applications do not reflect the main concepts from the domain.
Think about your menus in any desktop application: File, edit, view, insert, format, tools, window, help. Does the menu tell you what the application is about?
Think about these menus instead: Accounts, Payables, Receivables. Can you figure out what the application is about? If you said finance and internet banking, then you'd be right.
- ◀ Most applications are spread out over hundreds of screens all showing slightly different views and functions. You know your application is like this when users always say to each other 'where's that button again?'
- ◀ Most applications are designed by developers and BAs who have a limited, often wrong, understanding of the domain. Would you ever ask a novice to specify how the application should work to drive high performance?

We all know that most applications are challenged when it comes to usability and driving high performance. It just doesn't make sense to use them as a source of inspiration for your new application.

We all know that if you keep doing what you've always done, then you'll keep getting more of the same.

1.4. Getting accurate requirements

I think I've clearly illustrated that the current approach to gathering requirements is flawed. Poor requirements accounts for 50% of all defects, so why are we still using the same methods. UML doesn't help because you can't get the business to read and understand it to sign it off. More interviews don't help, because the developers and BAs do not have the necessary background to capture the user requirements properly. And simply asking people what they want is not the best source of information.

It should be clear that the current fact that requirements are changing throughout and after the development lifecycle is not because the requirements are actually changing (that is, people are suddenly doing sales or accounting differently this week than they did last week). It is because the **wrong requirements** were captured in the first place.

It is only by developing the application and demonstrating its user interface that users can then use it to see if it matches what they want. Because the answer will be invariably 'that's not what I asked for', you then need to iterate numerous times to attempt to arrive at an application that works.

You then have the issue that if the application matches how they are currently working, and that way is not efficient or effective, then all you are doing is bedding down incorrect practices.

So you can see that it's not a matter of changing requirements, it's a matter getting the requirements right through an iterative process, instead of using the right techniques by the right practitioners.

In Part 2 of this article, I'll outline the approach we take to capture user requirements correctly. As a preview, the three basic principles are:

- ◀ There is a clear distinction between business requirements and user requirements. They are not the same and capturing one does not mean you can infer the other.
- ◀ The analyst must have specialist skills to capture user requirements. These skills are in the social and management sciences, not in the computer and engineering sciences.
- ◀ User (and business) requirements must show a direct relationship with the business strategy and key performance indicators (KPIs). That is, when specifying how a user should be going about their task, you must demonstrate how it will positively contribute to achieving the KPIs

The fundamental technique comes from I/O psychology and is called job analysis. How job analysis works will be the subject of the follow-up article.

2. About the Author

Craig is the founder and Managing Director of The Performance Technologies Group (PTG Global), with over 15 years in user experience, user interface design and change management.

Craig runs the R&D function at PTG, having produced a number of world firsts including XPDesign – the first systematic methodology for user interface design and Certified Usable – the first guarantee for usability and user experience.

Craig has been the primary architect behind many of Australia's most popular websites including CBA, Virgin Blue and ASIC and works on cutting edge technologies such as touch, medical and special-purpose applications.

Craig holds a Masters qualification in organisational psychology, is a member of the APS and the APS College of Organisational Psychologists and is a Registered Psychologist in NSW. He is also an Associate of the University of NSW and Macquarie University.



Contact Craig on:

Email: craige@ptg-global.com

Phone: +61 (0)2 9251 4200

Mobile: +61 (0) 416 266 216

Address: Level 16, 207 Kent St, Sydney, NSW, 2000, Australia