

1. **Aligning people, business and technology: *Getting the requirements and user interface right to deliver technology that works for people and the business***

1.1. **The common complaint in IT projects: *Missing or poor requirements***

It's generally accepted that around 50 - 80% of all major IT projects fail in some significant way. They're usually over budget, very late, missing requirements or have poor user acceptance – sometimes all of the above.

Failures are often blamed on insufficient time or money, bad requirements, bad technology, not enough testing, stupid users or management without any vision. The 'solution' has been to throw more and more time, money and technology at it — but nothing seems to stem the failure rate.

These issues are merely symptoms, not the real problems. Why? It's all to do with the way IT projects are traditionally managed. This diagram summarises the traditional process:

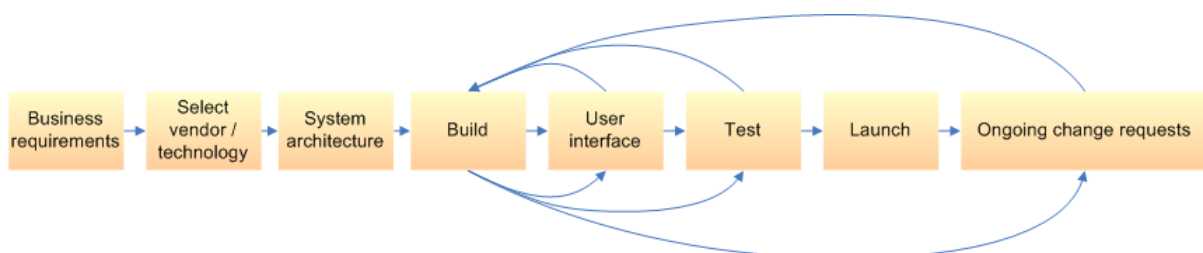


Figure 1. The traditional software development process, with mandatory iteration.

The process starts with capturing the business requirements, usually countless pages of written material, often including use cases.

A vendor is chosen against these requirements and a solution architected. The build starts and during this process, several things happen – developers fill in the gaps in requirements and create the user interfaces – all with little or no input from the end users.

The endless iteration starts here. The management and users review the screens and the first thing they say is 'that's not what I wanted'. The developers say 'Yes it is'. And around it goes.

So what's the real problem here? It's all to do with the requirements documentation. Because they're mostly text and system diagrams, everyone who reads it will have a different vision of the end product. And what 'vision' am I referring to? What you'll see on the desktop when it's all finished.

You're probably thinking to yourself: 'this is the way development has always been done'. Sure, it is. But think about this...when was the last time a building was constructed without first doing a detailed visual model?

IT is the only engineering discipline that does not have a clear and unambiguous blueprint of the end deliverable before the build starts. For IT, the blueprint is the user interface. It's the only thing you see and use when the project's finished – but it's always the last thing done in the development lifecycle.

Even if the entire user interface was completed before build commenced, there is still the problem of the initial requirements – if they're wrong or missing, it doesn't matter how good the user interface is, it still won't work for people and it won't deliver on the business objectives.

In the next section, I'll describe how to get the requirements right the first time using job analysis. After that, I'll introduce PTG Global's methodology for high performance user interface design, XPDesign, as a method to translate the requirements into design.

1.2. Using job analysis to get the requirements right

The main problem with existing requirements methods is that despite the (appropriate) involvement of end users, the techniques (i.e. interviews) typically used by BAs and developers cause the problem. It's here that the HR practitioner can have a significant effect on the quality of the requirements using job analysis.

Why use job analysis? If you agree that the application to be delivered is supposed to support people's work, then you need to understand the job people are performing.

Job analysis involves *key result areas* that organise *key tasks* into the broad areas of responsibility of the job. Finally, *key performance indicators* measure people's performance in the job.

Consider this example. Imagine the key tasks carried out by a sales person to sell a product to a customer...I'm sure you had many of these key tasks:

- Identify prospects
- Contact the prospect and meet them
- Identify needs
- Identify suitable products and services
- Demonstrate features and benefits
- Handle objections
- Close the deal
- Deliver the service or product
- Service the customer and cross / up sell

Now ask yourself this: what if they're selling loan products? Or cars? Or real estate? Do you think they all generally use the same steps? I would argue that, for the most part, they do.

What about the key tasks used 10 years ago? Or 20? Or 50? What about 50 years from now? Will people be selling things in generally the say way? Again, I would argue yes.

If you gave this list of tasks to a manager, developer, or end user, they would be quite clear on the activities of a sales person.

You now need to translate these requirements into a complete user interface design that represents the job. That's were XPDesign comes in.

1.3. Using XPDesign to translate the requirements into a high performance user interface

XPDesign is a systematic model-driven process to translate requirements to design, as shown in this diagram:

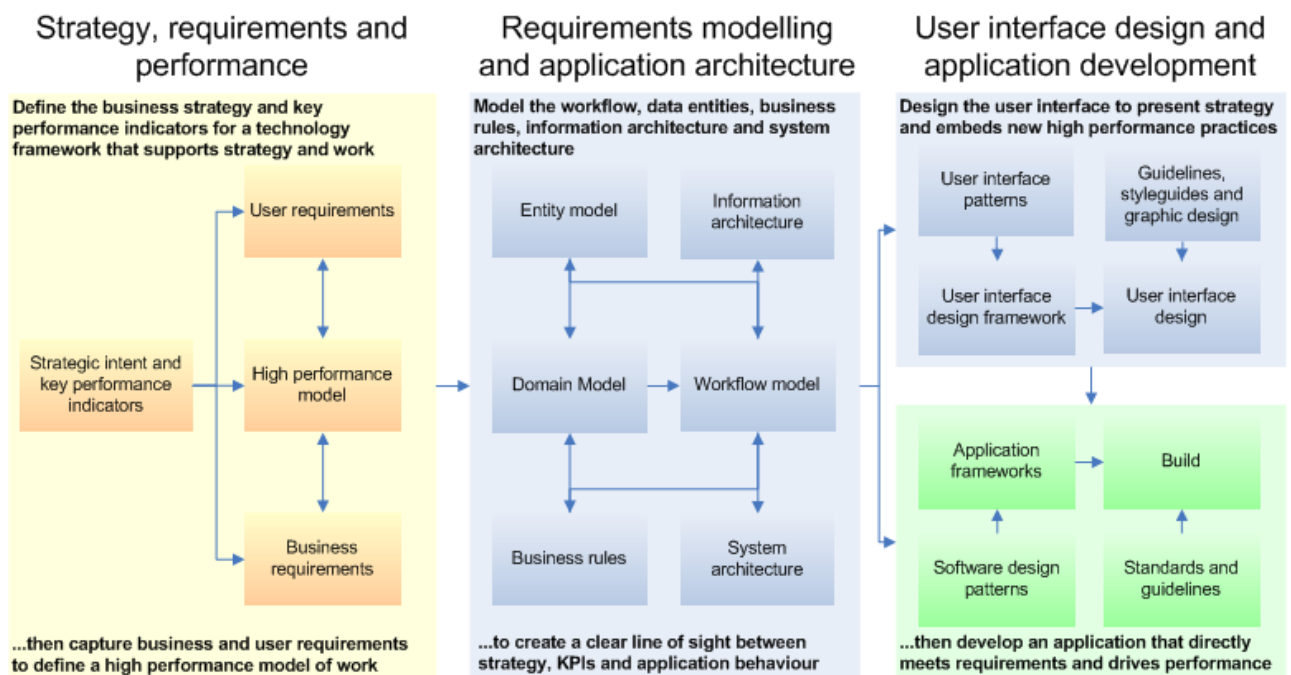


Figure 2. The XPDesign high performance user interface design methodology

The two most important models are objects and workflows, part of the user interface architecture – the foundation of the application and its behaviour - that produces a scalable and flexible framework for design.

The object model represents the data, people and things used in the interface to complete key tasks. The model is decomposed into deeper and more accurate representations of the data, until the database equivalent tables are reached. The key is that design is based on the objects people work with, not the hundreds of tables from the database. This is an example high level object model for a commercial lending application.

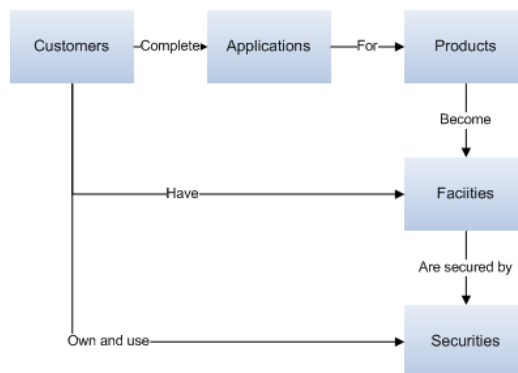


Figure 3. An example object model

The workflow model is a flowchart version of the job analysis. It is done at multiple levels, starting with a simple high level model and decomposed into detailed models containing low level database equivalent tasks (e.g. create customer'). It is important to start at a high level and not the detail, as it is easier to agree on simpler models and then drill down into the detail of each step. During design, sections of workflow are grouped to define screens. This diagram shows a high level workflow for commercial lending on the left.

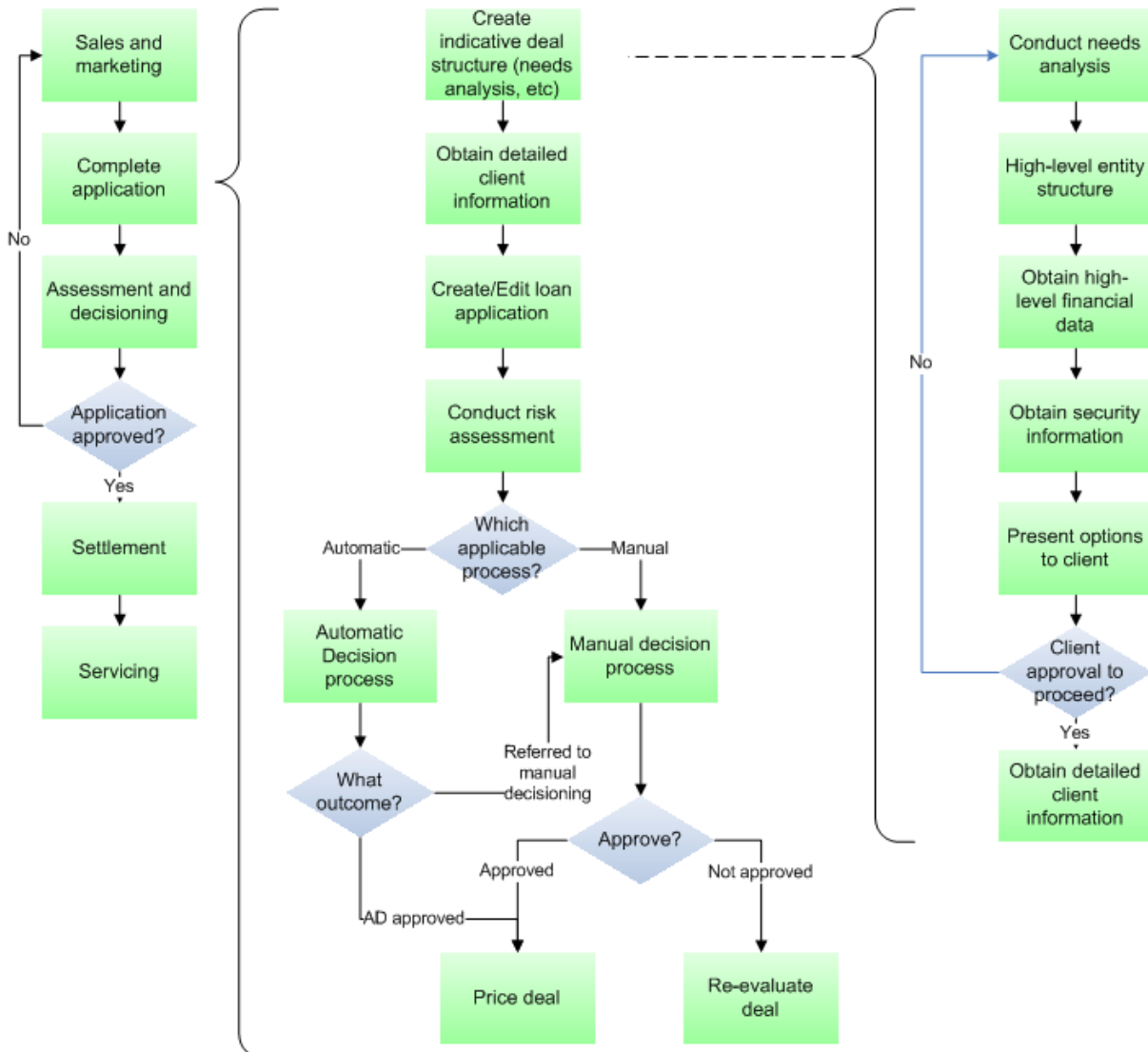


Figure 4. An example workflow model showing progressive decomposition of high level tasks into lower level tasks

In the example, the task 'Complete application' is then decomposed into its level 2 workflow. In that workflow, the 'Create indicative deal structure' task is decomposed into its level 3 workflow. Each low level task corresponds to a specific pattern, making user interface design a simpler and more consistent process.

Workflow modelling makes it easy to visualise what the application will do and is much easier to read than a thousand pages of use cases. The workflow can be tested against a range of 'what if' scenarios to ensure it handles 'real-life'. Unfortunately, we don't often see it done, or people start at the lowest level, where it's easy to argue in circles and become stuck on the minutiae.

This diagram shows how the models find their way into the user interface. This is the starting workbench for a commercial lender. The workflow model determines the activities being performed including working on tasks, applications and referrals. The object model defines the data needed, including customers and applications. User interface patterns deliver user-friendly data and functionality.

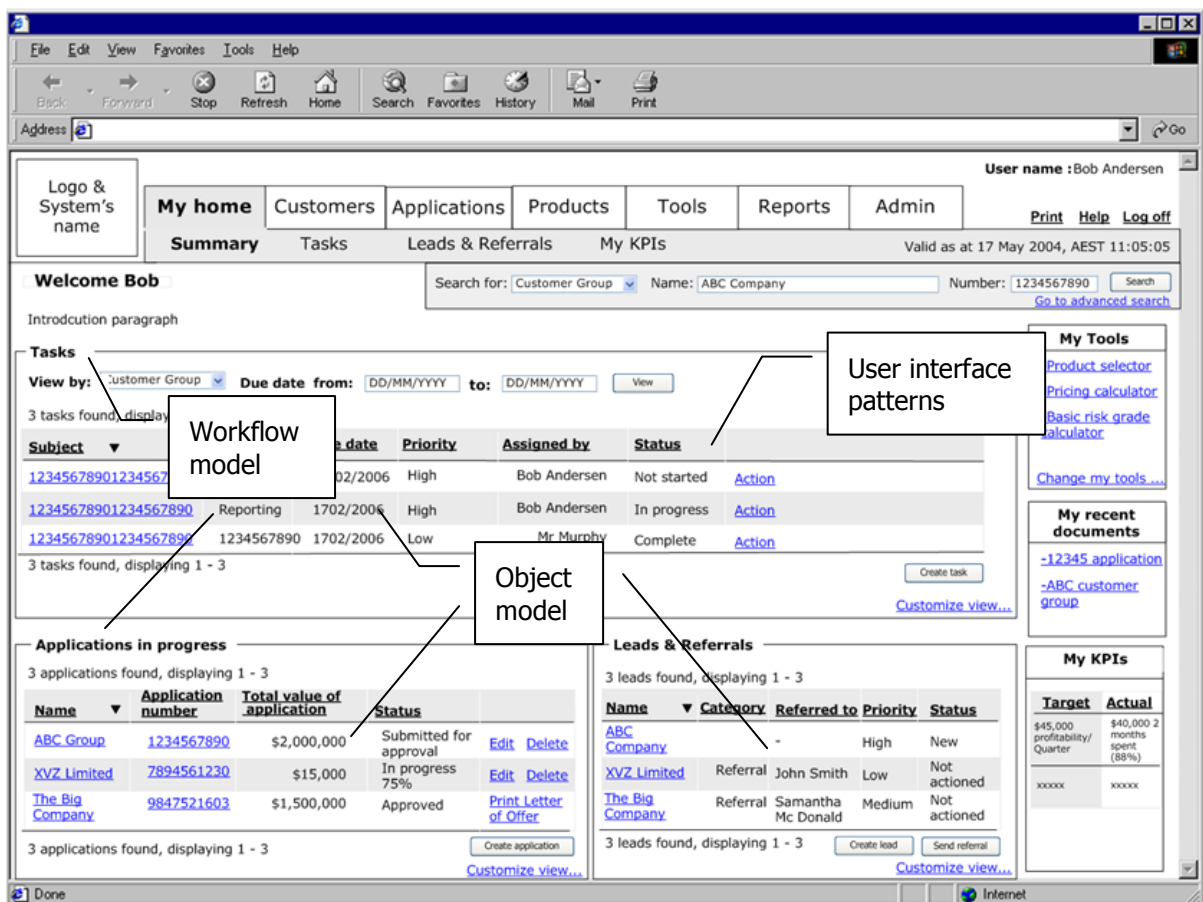


Figure 5. Integrating the object model and workflow model with user interface design

1.4. Putting it all together into the IT Blueprint

You need to do the interface at the right time in the software development lifecycle to deliver technology that works. This diagram illustrates the IT Blueprint:

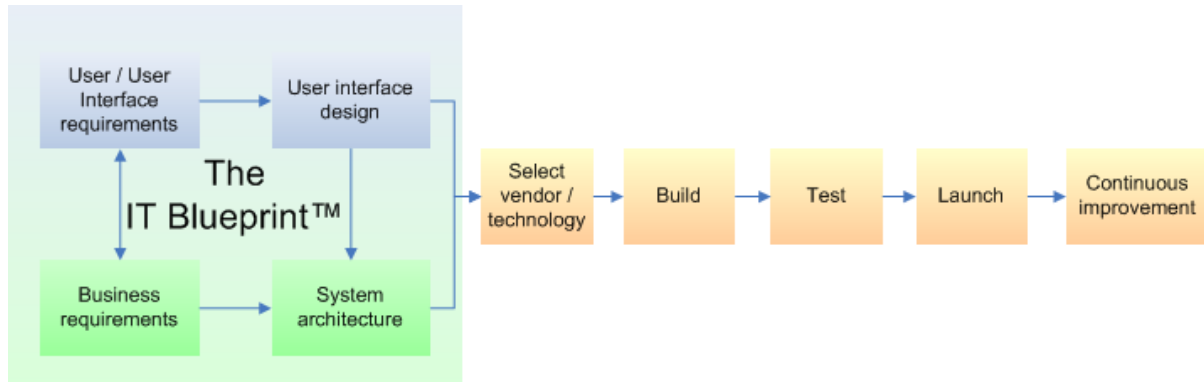


Figure 6. The IT Blueprint

The process uses exactly the same steps found in traditional IT development lifecycles, however, we have moved the user interface design to the beginning of the process. The complete user interface has now become part of the requirements, rather than a by-product of the build.

Imagine being able to see and test the entire user interface to determine if the application will meet requirements and deliver business benefit, before you buy or build anything. After all, the only thing people see and use is the user interface – it's the primary determinant of people's performance.

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