



## Reducing Time Spent on Requirements

Neville Turbit

### Breakdown of Time Spent

Recent research in the US indicated the time spent between

- Gathering requirements (Business Requirements & System Requirements) for a new system and
- Building and testing the system

was becoming more biased towards the requirement gathering. A range of languages were investigated and the results ranged from a 50:50 split for Assembler to a 90:10 split for Ada. In other words, for every week spent coding and testing in Ada, 9 weeks were spent gathering requirements.

<b>REQUIREMENTS</b> 50% to 90%	<b>CODE &amp; TEST</b> 10% to 50%
-----------------------------------	--------------------------------------

As languages become more sophisticated and develop drop and drag, point and click type interfaces, there is a real benefit to the developers but it doesn't have a big impact on the Business Users. Obviously the systems can be delivered sooner but the time the Business Users spend specifying the system is not significantly effected.

The trend to buy package solutions does not take away the issue of improving speed of delivery. If software is to be written from scratch or bought, the requirements still need to be developed.

In one sense, if you have a choice between cutting the requirements gathering process by a month on every project or, with the same effort, cutting coding and testing time by a month, the obvious focus would be on requirements. Requirements are needed whether you write your own software or buy it.

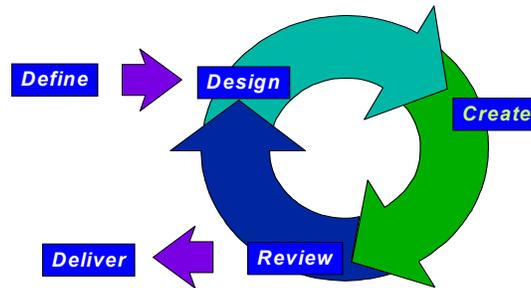
### **RAD (Rapid Application Development)**

Over the years there have been many advances made in the quest to reduce the time taken to build applications. RAD (Rapid Application Development) techniques have been around for some time and range from software development tools to prototyping techniques. Their focus is on the building of the system - the coding. Better tools to build databases, code generators, documentation tools all focus on the build component of the exercise.

Automated testing tools have made great advances in speeding up the test times. With one exception - prototyping - all neglect what is becoming the largest part of the process - requirement gathering.

## Prototyping

Prototyping is the building of a part of the system from less detailed requirements than would normally be the case. Instead of completing a highly detailed specification and disappearing for six months, the IT Department delivers a mock up of the system for the users to review after say one month. Over the following months, further refinements are made and the system is built with regular input from Business Users.



Whilst Prototyping can reduce the overall development time, there is usually not a saving in the time Business Users spend translating their views, into something the IT Department can build. The lack of detail up front takes less time, but the review of prototypes during development takes time. The net result is that Business Users end up spending the same amount of time over a longer period.

The advantage in prototyping is the influence the Business can have on the final deliverable. The system is not built to a specification. It is built based on an unraveling of the ideas in people's minds.

This is the benefit of prototyping. The danger is that the prototyping goes on forever seeking the perfect solution. There are ways to handle the endless prototype but that is a different topic.

## Requirement Gathering

We have identified four major causes of difficulty in gathering system requirement and business requirements

### 1. Difficulties in Visualization

If we were asked to sit down at a desk, given a pad of paper and asked to write down a detailed specification for a dream home, it would be a daunting task. On the other hand, if we were asked to give a big picture view (e.g. the general layout) and were asked back when the frame was built to make final decisions, or modify, the placement of walls, windows etc. the result will be more to our liking. It is a more natural way to work and is the easier way to create a specification from a conceptual view.

Sometimes in System Development it is an easy task to create a specification. A data conversion is simply a list of where the existing information goes in a new system. It lends itself to a specification type approach. Most new systems however do not mimic existing systems hence there is a visualization leap required. People are asked to design something they have unclear ideas about.

If an Engineer is required to design a bridge, they know the starting and finish points, the loads, the soil composition etc. and, because of their technical training, can easily specify the required solution.

A Business User who needs to specify a new system usually has no clear view of how the new system will mesh with, or in fact replace the existing environment. Add to this the fact that the Business person usually has little understanding of the technical environment or requirements and it is a little like asking the person who will drive over the bridge to design the bridge.

## **2. Difficulties in gaining access to people**

The second problem is the way IT projects are treated from a Business perspective. Often it is seen as a “second job” which has to be done on top of ongoing duties.

“I spend 10 hours a day as the Supervisor in this department, and now they want me to find time to help IT specify a replacement system!”

No wonder IT end up making many of the decisions Business should be making. Business are often unavailable, or don't have the time to think it through. Decisions are made on the fly by part timers on the project. Alternatively IT give up and make the decisions themselves.

## **3. Difficulties in Compromise**

A third problem flowing from this is the issue of compromise. In any development, compromises will need to be made. For example:

- All the information required will not fit on the report
- You have to enter certain information if you want a particular output.

Many of these compromises are made by IT alone because Business Users are not around at the time. The end result is that Business are delivered a system which they say is not what they want.

“Why is this information missing from the report?”

“I don't always have that particular information when I am entering the data but the system will not let me proceed unless I fill in this field.”

If Business were involved in the compromise, they would be more accepting of the result. They would be involved in reaching a solution rather than not understanding the context in which the compromise was made.

## **4. Difficulties in reaching Decisions**

The fourth factor is the lack of skill in extracting information and reaching decisions. Very little training takes place in how to get information on a topic from one person much less a group. We all attend meetings where a specific topic is to be discussed and the meeting ends up going around in circles and off on tangents without actually resolving the issue. There are no training courses to teach focus on topics and quick resolution of information gathering sessions.

Sometimes the purpose itself is not clear. Add to this the complication in System Development, that what is being discussed is being created - there may not be an existing model to work from - and the difficulty is increased substantially. There may not even be a right answer. There are usually a range of alternatives, several of which may be equally acceptable. Decisions can be difficult to reach.

## US Research

Recent research in the USA of 8,500 projects showed 31% were never completed. All indications are that Australia would follow that trend. The reasons projects were abandoned were:

- Lack of user input - 12.8%
- Incomplete requirements & specs-12.3%
- Changing requirements & specs - 11.8%
- Lack of executive support - 7.5%
- Technological incompetence - 7.0%
- Lack of resources - 6.4%
- Unrealistic expectations - 5.9%
- Unclear objectives - 5.3%
- Unrealistic timeframes - 4.3%
- New Technology - 3.7%
- Other - 23.0%

The first three reasons all relate to ...”we don’t know what we are building”... Add to this “Unrealistic expectations” and “Unclear objectives” and about half of the reasons relate to ...”we don’t know what we are building”... Because of the lack of clarity, the projects failed.

If we look at the remaining 69% that were completed

- 68% were more than 50% over time
- 53% more than 50% over budget.

The reasons were much the same:

- Incomplete requirements - 13.1%
- Lack of user involvement - 12.4%
- Lack of resources - 10.6%
- Unrealistic expectations - 9.9%
- Lack of executive support - 9.3%
- Changing requirements & specifications - 8.7%
- Lack of planning - 8.1%
- Didn’t need it any longer - 7.5%
- Lack of IT Management - 6.2%
- Technology Illiteracy - 4.3%
- Other - 9.9%

Lack of resources was as much Business as Technical resources. Once again the results reflect the difficulty in getting Business Requirements sorted out. The project

didn't fail because they couldn't cut the code. The failure was in understanding what to build.

The end result of the research was a guide to the weightings for success of IT Projects. The weightings were out of 100.

### **Success Factors (weighted)**

- User Involvement - 19
- Executive Management Support - 16
- Clear statement of requirements - 15
- Proper Planning - 11
- Realistic Expectations - 10
- Smaller Project Milestones - 9
- Competent Staff - 8
- Ownership - 6
- Clear Vision & Objectives - 3
- Hard Working Focused Staff - 3

Basically you need

- The Business to specify what they need
- Have the right resources
- Have Executive Support
- Plan it properly in small chunks with regular checkpoints
- Provide teams with responsibility and accountability to achieve a realistic, common objective.

### **Past Attempts to Improve Requirement Gathering**

There have been two major events over the years that have been significant attempts to speed up requirement gathering. They are

- The creation of the role of the Business Analyst
- JAD or Joint Application Development.

To understand how the BA role evolved, it is important to look at how System Development evolved.

### **The Three Generations of System Development**

System Development has been through two generations. There is the start of a third generation but that will be discussed later in the paper.

#### **Generation 1. Direct Contact**

The first was when Business and IT worked directly together to develop requirements. In fact many projects still use this first generation approach and it works in some situations.



The limitation with this approach is that Business people sometimes think the IT people are speaking another language. The message gets lost in the jargon.

IT on the other hand are listening without having the background to understand the context in which the Business takes place. They may not be aware of the degree of effort required to train temporary staff, or the drain on resources this causes when discussing what they see, as oversimplification of input screens.

## Generation 2. Business Analyst

The second generation was the creation of the Business Analyst. The BA was initially a person who was highly skilled in the Business area and given enough training in IT to be able to translate the requirements from a Business environment to a Technical environment.



This was initially highly successful. The reason was, the BA knew the requirements better than the User doing the specification. The BA provided most of the requirements out of their own head, and topped them up with the User's input. As time went on, the BA became a position recruited from outside the company. BA's became technicians in their own right, and the role became one of translation and guidance rather than providers of the information.

Suddenly the users knew more about the requirements than the BA's. The potential problem can be that the Users tell the BA who tells the IT Department. Sometimes, the message gets distorted.

## JAD

JAD was a technique developed by IBM in the 70's. JAD stands for Joint Application Development or Joint Application Design. It is a technique to gather a group of Users together in a room with an independent Facilitator and make decisions regarding what is required.

The key components are;

- There is a process which is managed by the Facilitator
- The focus is on gaining consensus regarding the outcomes
- The outcomes are the decisions made regarding the proposed system

JAD is not just limited to System Development. It is a valid technique to use for any situation where there is a need to extract information or resolve issues with a group of people. The team dynamics come into play in a JAD session and consensus is more easily obtained.

The key to JAD is to establish a process *before* the meeting begins. If a framework exists, people know what to do when a situation arises that causes difficulty. For

example one rule or guideline is the “15 Minute Rule”. If an issue is discussed for more than 15 minutes, it is noted down as an issue to be resolved outside this meeting, and the group moves on to discuss issues that can be resolved. It could equally be a “5 Minute Rule” or “25 Minute Rule” depending on the environment and complexity of the subject being discussed. If the purpose is to reach decisions, there is no point in getting stuck on the most difficult and neglecting the ones which can be made quickly.

Another technique is for the team to appoint a “Tie Breaker” before the meeting commences to make a decision where the meeting is split over an issue.

IBM made a valid attempt to improve the speed of generating the requirements, and forcing the involvement of Business in the decision making. To a large extent it worked. JAD lost momentum during the 80’s due to the increasing complexity of the ground rules. The process became too complex. There are good ideas in JAD but the whole process needs to be kept simple.

## The Way Forward

To summarize, the problems identified to date can be distilled down to:

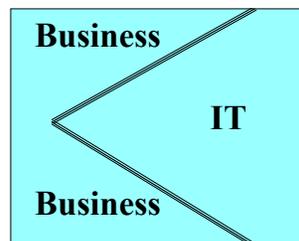
- Lack of **Business involvement**
- Lack of a **process to force decisions**
- Reliance on developing an accurate and detailed specification for the system, rather than **applying an evolutionary approach**

### Business Involvement

The third generation of System Development is the way to ensure Business Involvement.

### Generation 3 - Team Based

The third generation is the flexible team. The team consists of a changing mix of Business and IT people. The core stays the same but people come and go into the team depending on the status. In the early days of Business Specifying what they need, it is heavily weighted with Business people. As development takes place, the weighting swings to IT. At implementation, the weighting swings back to a more balanced mix.



In order for this to succeed, there needs to be a communication layer in place. There needs to be tools and techniques used to enable the two parties to communicate and understand each other’s point of view. Using the house building analogy again, you can go to a paint store and look on a PC at room layouts with different coloured walls. It is a way for the home owner (or user) to communicate with the painter (or technician).

The three generations of IT Development have, and can work successfully in *some* situations. It is the third generation however that has the most chance of success in *most* situations.

Before the project is started, Business commitment to participate fully in the process needs to be obtained. With this in place a team or teams of Business and IT people are set up and provided with the tools needed to communicate and get the job done.

Teams don't just happen. A group of people do not constitute a team. There needs to be significant work done to set up the correct structure and environment for a team to work.

The way teams operate needs to be based on the result of research, psychology and experience. Teams need to be initiated in an environment that gives them every chance for success. In fact even making them able to judge their success or lack of it is part of the environment. In addition you need to provide:

- Roles and Responsibilities for everyone so each person knows what is expected of them, and how they interact
- A clearly defined goal to be achieved by the team
- Work carried out in short intensive bursts to retain focus. Typically a team works to achieve a goal in a month
- An understanding that they are working to fixed timeframes. The end date never changes.
- A problem resolution mechanism to avoid the “what do I do now something has gone wrong” situation
- The team creates and owns the plan of what they are to do. Responsibility matched with the Authority to take decisions and make things happen.
- Milestones every few days to ensure they are on schedule
- A range of tools to help them gather the information they need (e.g. JAD, Data Modelling Tools, etc.)
- Training before they start to ensure everyone knows what they are doing, how they are going to do it, and when it is to happen.

### **Process for Decision Making**

The best tool available to force decision making is the JAD session. Not traditional JAD but the good points of the traditional approach, blended with some other innovative ideas. The result is highly productive, facilitated sessions to generate requirements.

In themselves, JAD sessions are much faster to produce results than traditional methods. With good facilitation the time taken can be halved again. The difference between a well facilitated JAD and a poorly facilitated JAD can be double the time to get the same result. This has been established in our own research.

Success breeds success. If Business Users believe the process of JAD is a highly productive exercise for them, they will be even more supportive of using the technique in the future and more enthusiastic to participate in development projects.

## **Evolutionary approach for Requirements**

To assist the process of allowing requirements to evolve, you need to use the 80:20 rule. Spend 20% of the time to get 80% of the requirements. Later you come back and spend 20% of the time to get 80% of the remaining requirements and so on, until you have the full details. The break allows people to unravel the issues in their own mind and think things through.

An example may be a simple system to provide basic customer data. The traditional approach says, take a stab at the time and cost, get approval, and then specify in detail the requirements.

The 80:20 approach is to identify there are 3 screens. The layout of those screens and all the detailed information is not reviewed. This gives us enough information to size the development and produce a sensible Business Case based on a costing that is not plucked out of the air. We don't need all the specification to make a realistic estimate of cost.

When approved, we can look at the general content of the screen. By that time people will have thought about it a bit more and have a better idea of what is required. (e.g. we need to record first name, surname, have 3 lines to enter the address, home, home fax, office, office fax and mobile numbers etc.)

Finally in a prototyping approach to development we can fine tune where the information is on the screen, add more if necessary, move it around, create business rules etc. All this is done in a controlled prototyping approach.

It may well be that a few months pass before the detail is added to the big picture created in the first phase. A solution is allowed to evolve and the team discuss the progression formally and informally over the months. There is still time, right up to development, to add that idea you had at 3 am in the morning.

By encouraging evolution you encourage creativity and the design of better systems. If, on the other hand, it is understood the specification cannot be changed, creativity is stifled and frustration sets in.

You need to use this broad and constantly narrowing approach. You do not take the first area, drill down to provide a detailed specification with enough detail to start cutting code, before you start on area two. You peel the onion layer by layer.

## **Other Aids**

Whilst we have covered the major contributors to reducing the time to gather requirements, there are some other tools that will aid the gathering of requirements.

### **Break it Down**

When General Norman Shawartzakoff was asked about the logistical effort required for Desert Storm he said

“It is like eating an elephant. One bite at a time.”

Every project needs to be broken down into small chunks or phases that people can get their mind around. Small bites, and intensive chewing.

### **Templates**

Much of the time taken to produce a document such as a Business Case is in deciding the structure. By providing templates for an organisation, with each section of the

template explained and examples given, the wheel does not have to be re-invented. It also provides better consistency in the output. The quality of the deliverables is higher as there are models to work from. Training is easier as it can be clearly seen what has happened in the past.

### Quality Assurance

QA needs to be carried out on the deliverables from each team. It needs to be an independent inspection of the deliverables in order to maintain quality and avoid the informal peer group review that typifies most output.

The inspection should be a short time to find big problems rather than the latter. The process is minimum effort for maximum return to find the “show stoppers” rather than the typos. It should be non threatening and ensure the output meets organisational standards.

### Planning

Planning needs to happen before each phase commences. In fact, if the planning does not take place, the phase should not commence. The typical “Lazy S” curve where productivity falls in the early stages as a team works out what it has to do and how to go about it, and picks up later, is avoided. The team hits the ground running when the phase begins.

### Repeatability

Doing things the same way provides efficiency and control. Everyone understands what they are doing and results are comparable. People can be interchanged between projects and can be up and running faster on the next project. It also provides the opportunity to constantly improve the process as lessons are learnt on each project and addressed in a formal Project Review. Training is easier as there are real life models within this organisation to learn from.

## Evaluating an Organisation

In order to evaluate how well your organisation performs, you might like to answer the following 12 questions.

Score	1	2	3
How would you rate the relationship between your IT Department and the Business areas.	IT under siege	Patchy	Good
Do the Business perceive they are getting value for money from IT	No	Sometimes	Yes
Are Projects always on time	Rarely	Sometimes	Always
Are Business available to participate in requirement gathering	Rarely	Sometimes	Always
Is there scope for requirements to be generated more quickly	Yes	Perhaps	No
Is there an attitude that change should be embraced	No	Occasionally	Yes

Score	1	2	3
Are decisions regarding requirements made quickly	No	Occasionally	Yes
Are there standard templates for project documents to stop re-inventing the wheel	No	Some	Yes
Is there a QA process in place for phase deliverables	No	Informal Process	Formal Process
If a Project Manager leaves, is there a repeatable process in place which enables the project to continue without disruption	No		Yes
Do project have Executive Support when it is needed	No	Sometimes	Always
Do the project teams produce plans they actually achieve	No	Sometimes	Always

If your organisation scored more than 30 you have a highly successful IT area. Few organisations honestly evaluating themselves, rate over 30. Perhaps the IT department and the Business areas might like to carry out the rating independently and compare notes at the end.

If your rating is below 30, there are steps you can take to lift the rating. Treat it as a benchmark and review it in 12 months based on improvements you introduce during the year. Prove to the organisation the IT area can lift it's game and move towards a service oriented asset to the organisation.

## Summary

The place where time can be saved in a System Development project is in the gathering of requirements. If marginal improvements can be made in requirement gathering, the maximum impact will occur in overall times. Development and testing has already been squeezed and there is not much more scope left to improve in that area.

Requirements are still not particularly well handled and usually treated as a contractual specification which can only be changed with a great deal of pain. They should be fluid snapshots of requirements that are allowed to grow and evolve. The Project Team should be looking for, and taking on changes to encourage creativity.

The traditional way organisations go about talking to people, and the process they use to gather requirements and make decisions has much scope for improvement. A modified version of JAD is the best tool we have to make this happen in a faster more co-operative manner. JAD will improve productivity noticeably when used with a good Facilitator.

The key to all good specification is to be Business Driven. To make this happen we need the appropriate tools and techniques to facilitate communication and the

gathering process. More importantly we need to gain Business commitment before we start.

One fact often overlooked is that Business will spend the same time on any project. Sometimes it is spent up front getting it right; sometimes down the end fixing it up. Where would they rather spend the time?

Given these issues are addressed we can build systems faster, and for less cost. They will be what the Business wants, and incorporate many of the ideas that currently are suppressed or discarded. The working environment will be more co-operative between Business and IT, and quality higher.

Using this approach is not a guarantee of all Systems on time and budget any more than a good accounting system is a guarantee of a solid financial company. The value it can bring is that it creates the environment where success is more probable. Given a 31% abandonment rate for projects, anything that helps substantially, to reduce the risks is a necessity.

Neville Turbit has had over 15 years experience as an IT consultant and almost an equal time working in Business. He is the principal of Project Perfect. Project Perfect is a project management software consulting and training organisation based in Sydney Australia.

Project Perfect sell “Project Administrator” software, which is a tool to assist organisations better manage project risks, issues, budgets, scope, documentation planning and scheduling. For more information on Project Administrator or Project Management visit [www.projectperfect.com.au](http://www.projectperfect.com.au)