

# A Project Management Primer

or “a guide on how to make projects work”

by Nick Jenkins



©Nick Jenkins, 2005

<http://www.nickjenkins.net>

This work is licensed under the Creative Commons (Attribution-NonCommercial-ShareAlike) 2.5 License.. To view a copy of this license, visit [<http://creativecommons.org/licenses/by-nc-sa/2.5/>]; or, (b) send a letter to “Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA”.

In summary - you are free: to copy, distribute, display, and perform the work and to make derivative works. You must attribute the work by directly mentioning the author's name. You may not use this work for commercial purposes and if you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one. For any reuse or distribution, you must make clear to others the license terms of this work. Any of these conditions can be waived if you get permission from the copyright holder. Your fair use and other rights are in no way affected by the above. Please see the license for full details.

# Table of Contents

<b>INTRODUCTION.....</b>	<b>3</b>
<b>BASIC PRINCIPLES.....</b>	<b>4</b>
<b>Ten Axioms for Success.....</b>	<b>4</b>
<b>Scope Triangle.....</b>	<b>6</b>
<b>The Critical Path.....</b>	<b>7</b>
<b>The Mythical Man Month.....</b>	<b>8</b>
<b>SCOPE .....</b>	<b>10</b>
<b>Scope, Visions and Goals.....</b>	<b>10</b>
<b>A Decent Proposal.....</b>	<b>12</b>
<b>Requirements.....</b>	<b>15</b>
<b>Requirements capture.....</b>	<b>17</b>
<b>Documenting Requirements.....</b>	<b>19</b>
<b>Traceability.....</b>	<b>22</b>
<b>PLANNING.....</b>	<b>23</b>
<b>The Purpose of a Project Plan.....</b>	<b>23</b>
<b>The Fine Art of Scheduling.....</b>	<b>24</b>
<b>Costing and Budgeting.....</b>	<b>29</b>
<b>Risk Management.....</b>	<b>31</b>
<b>Change Management .....</b>	<b>33</b>
<b>EXECUTION.....</b>	<b>35</b>
<b>Staying on track.....</b>	<b>35</b>
<b>Managing People.....</b>	<b>36</b>
<b>IMPLEMENTATION.....</b>	<b>39</b>
<b>REVIEW .....</b>	<b>42</b>
<b>GLOSSARY.....</b>	<b>43</b>

# Introduction

Many projects fail because of the simplest of causes. You don't have to be a genius to deliver a project on time, nor do you have to be steeped in a mystical project management methodology to be a project manager. If an averagely competent person can't deliver a project successfully after reading this then I will run buck naked through Times Square on my 75<sup>th</sup> birthday. See if I don't!

That reminds me of a joke...

A tourist walked into a pet shop and was looking at the animals on display. While he was there, another customer walked in and said to the shopkeeper, "I'll have a C monkey please." The shopkeeper nodded, went over to a cage at the side of the shop and took out a monkey. He fitted a collar and leash, handed it to the customer, saying, "That'll be £5,000."

The customer paid and walked out with his monkey. Startled, the tourist went over to the shopkeeper and said, "That was a very expensive monkey. Most of them are only a few hundred pounds. Why did it cost so much?" The shopkeeper answered, "Ah, that monkey can program in C - very fast, tight code, no bugs, well worth the money."

The tourist looked at a monkey in another cage. "Hey, that one's even more expensive! £10,000! What does it do?"

"Oh, that one's a C++ monkey; it can manage object-oriented programming, Visual C++, even some Java. All the really useful stuff," said the shopkeeper.

The tourist looked around for a little longer and saw a third monkey in a cage of its own. The price tag around its neck read £50,000. The tourist gasped to the shopkeeper, "That one costs more than all the others put together! What on earth does it do?"

The shopkeeper replied, "Well, I haven't actually seen it do anything, but it says it's a project manager".

## Ten Axioms for Success

To help you get started here's ten (self evident) truths :

### I. Know your goal

It sounds obvious but if you don't have an end-point in mind, you'll never get there. You must be able to clearly state the goal of your project so that anyone can understand it. If you can't adequately describe your goal in a single sentence then your chances of achieving it are pretty slim.

### II. Know your team

Your team is the most important resource you have available and their enthusiastic contribution will make or break your project. Look after them and make sure the team operates as a unit and not as a collection of individuals. Communications are vital! Invest time in promoting trust and ensuring that everyone knows what they have to contribute to the bigger picture. Dish out reward as well as criticism, provide superior working conditions and lead by example.

### III. Know your stakeholders

Spend time with your stakeholders. Stakeholders will either contribute expert knowledge to the project or will offer their political or commercial endorsement which will be essential to success. Shake hands and kiss babies as necessary and grease the wheels of the bureaucratic machine so that your project has the smoothest ride possible.

### IV. Spend time on planning and design

A big mistake traditionally committed on projects is to leap before you're ready. When you're under pressure to deliver, the temptation is to 'get the ball rolling'. The ball however, is big and heavy and it's very, very difficult to change its direction once it gets moving. You need to spend time deciding exactly how you're going to solve your problem in the most efficient and elegant way.

### V. Promise low and deliver high

Try and deliver happy surprises and not unpleasant ones. By promising low (understating your goals) and delivering high (delivering more than your promised) you :

- Build confidence in yourself, the project and the team
- Buy yourself contingency in the event that things go wrong
- Generate a positive and receptive atmosphere

Consider this : if you finish early everyone will be happy; if something goes wrong you might still finish on time and everyone will still be happy; if things goes really badly you might still not deliver what you anticipated but it will still be better than if you over-promised!

## **VI. Iterate! Increment! Evolve!**

Most problems worth solving are too big to swallow in one lump. Any serious project will require some kind of decomposition of the problem in order to solve it. This works but only with close attention to how each piece is analysed and resolved and how the whole fits together. Without a systematic approach you end up with a hundred different solutions instead of one big one.

## **VII. Stay on track**

Presumably you have an end goal in mind. Maybe it's your job, maybe your business depends upon it or maybe you're going to revolutionise the world with the next Google, the next World Wide Web or the next Siebel/SAP/Oracle.

If this is the case you need to work methodically towards a goal and provide leadership (make decisions). This applies whether you're a senior project manager running a team of 20 or you're a lone web developer. You need to learn to use tools like schedules and budgets to keep on track.

## **VIII. Manage change**

We live in a changing world. As your project progresses the temptation to deviate from the plan will become irresistible. Stakeholders will come up with new and 'interesting' ideas, your team will bolt down all kinds of ratholes and your original goal will have all the permanence of a snowflake in quicksand. Scope creep or drift is a major source of project failure and you need to manage or control changes if you want to succeed.

This doesn't imply that there should be single, immutable plan which is written down and all other ideas must be stifled. You need to build a flexible approach that allows you to accommodate changes as they arise. It's a happy medium you're striving for - if you are too flexible your project will meander like a horse without a rider and if you are too rigid your project will shatter like a pane of glass the first time a stakeholder tosses you a new requirement.

The best way to handle this is to have a plan, to update it regularly and make sure everyone is following it and pointing in the same direction.

## **IX. Test Early, Test Often**

Project usually involve creative disciplines loaded with assumptions and mistakes. The only way to eliminate errors is through testing. Sure you can do a lot of valuable work to prevent these mistakes being introduced, but to err is human and some of those errors will make it into your finished product code. Testing is the only way to find and eliminate errors.

## **X. Keep an open mind!**

Be flexible! The essential outcome is delivery of the finished project to a customer who is satisfied with the result. Any means necessary can be used to achieve this and every rule listed above can be broken in the right circumstances, for the right reasons.

Don't get locked into an ideology if the circumstances dictate otherwise.

Don't get blinded by methodology.

Focus on delivering the project and use all the tools and people available to you. Keep an eye on the schedule and adjust your expectations and your plan to suit the conditions. Deliver the finished product, promote its use, celebrate your success and then move on to the next project.

# Scope Triangle

Called the 'Scope Triangle' or the 'Quality Triangle' this shows the trade-offs inherent in any project.

The triangle illustrates the relationship between three primary forces in a project. Time is the available time to deliver the project, cost represents the amount of money or resources available and quality represents the "fit-to-purpose" that the project must achieve to be a success.



In reality the normal situation is that one of these factors is fixed and the other two will vary in inverse proportion to each other. For example "Time" is often fixed in a project and the "Quality" of the end project will depend on the "Cost" or resources available. Similarly if you are working to a fixed level of "Quality" then the "Cost" of the project will largely be dependent upon the "Time" available (if you have longer you can do it with fewer people).

A phenomenon known in project management circles as "scope creep" can be linked to the triangle too. Scope creep is the almost unstoppable tendency a project has to accumulate new functionality. Some scope creep is inevitable since early on, your project will probably be poorly defined and will need to evolve. A large amount of scope creep however can be disastrous.

When the scope starts to creep new functionality must be added to cover the increased scope. This is represented by the quality arm of the triangle, representing the ability of the 'product' to fulfil users' requirements. More requirements fulfilled = a better quality product.

In this situation you have three, and only three options :

1. Add time – delay the project to give you more time to add the functionality
2. Add cost – recruit, hire or acquire more people to do the extra work
3. Cut quality – trade off some non-essential requirements for the new requirements

If the art of management lies in making decisions, then the art of project management lies in making decisions quickly! When faced with such a dilemma you should not hesitate to take one of the three options listed above. Delaying raises the risk of your project failing.

The astute reader will be wondering to themselves what happens when two of the points are fixed. This is when it gets really interesting. Normally this occurs when costs are fixed and there is a definite deadline for delivery, an all too familiar set of circumstances. Then, if the scope starts to creep you are left with only one choice – cut functionality. This is more common than you might think, in fact it's more common than not!

Cutting functionality may seem a drastic measure, but an experienced project manager will happily whittle away functionality as if they were peeling a potato. As long as the core requirements remain, everything will be fine. Additional functionality can always go into "the next project", if you don't deliver the core functionality, there won't be a next release.

A really experienced project manager might even pad his project with a little superfluous functionality that could be sacrificed when the crunch comes (but you didn't hear it from me!).

A poor project manager will see the scope triangle as a strait-jacket by which their project is irrevocably restrained. A better project manager will make better use of one or more of the axes and will recognise when they need to shift the emphasis in the project to one of the other axes. The best project managers will juggle all three like hot potatoes and will make decisions every day which effectively trade-off time vs quality vs resources.

# The Critical Path

Another important concept in planning projects is that of the critical path. If a project consists of a set of tasks which need to be completed the critical path represents the minimum such set, the critical set. This might seem to be a contradiction since surely completion of all tasks is necessary to complete a project; after all, if they weren't necessary they wouldn't be part of your project, would they?

The critical path represents not the ideal set of tasks to be complete for your project, but the minimum set. It is this path that you must traverse in order to reach completion of your project on time. Other tasks while important to overall completion do not impact upon the final delivery for the project. They can therefore be rescheduled if time is tight or circumstances have changed. Tasks on your critical path however will affect the delivery time of the project and therefore should only be modified in extremis.

In the following example the critical path is represented in bold. In order to complete my project of cooking breakfast I have to go through the steps of frying bacon and sausages and scrambling eggs.

The tasks "make toast" and "wash plates", while important, are not time-dependent or as critical as the other three tasks. I can move either of those tasks but if I try to move anything on the critical path its going to delay the project.

Ideally I'd like to have toast with my breakfast but a) it's not essential and b) it doesn't matter where in the process it happens. If I make toast before or after scrambling my bacon, it makes little difference to the overall result.

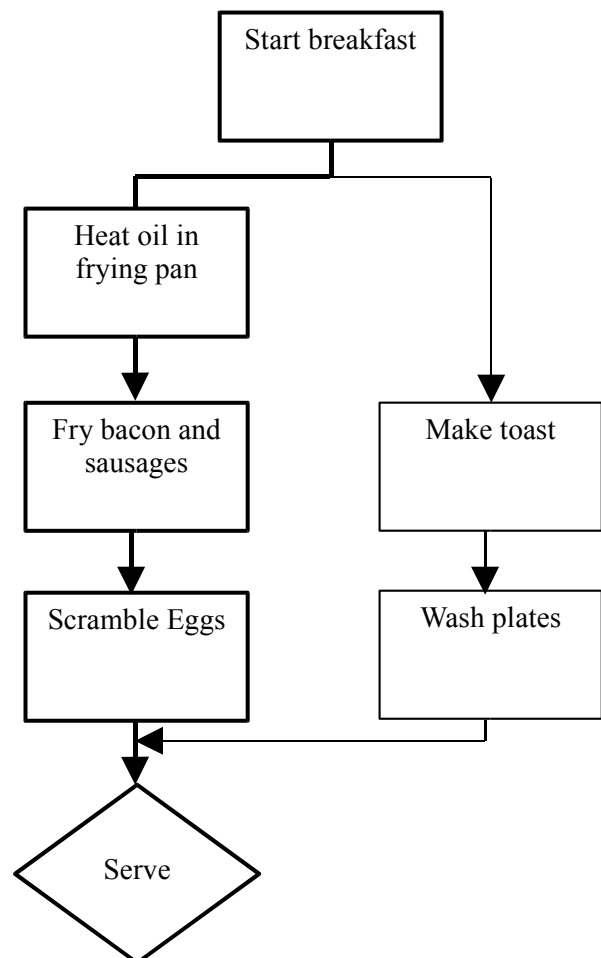
On the other hand I can hardly fry my bacon before the oil is hot, nor can I scramble my eggs before frying my bacon (they'd turn to glue).

The critical path represents the critical sequence of events which *must* occur if I want to successfully complete my project.

Normally major milestones will be represented on the critical path and they will often occur when different threads of the project come together.

For example in the diagram to the right my only milestone is when I serve the completed breakfast. At this point I will have finished my preparations and completed everything on both tracks.

If I suddenly discovered I was late for work I could cheerfully discard the optional "toast" component of my project, take the critical path instead and still achieve my original milestone of delivering breakfast (and even make it to work on time!).



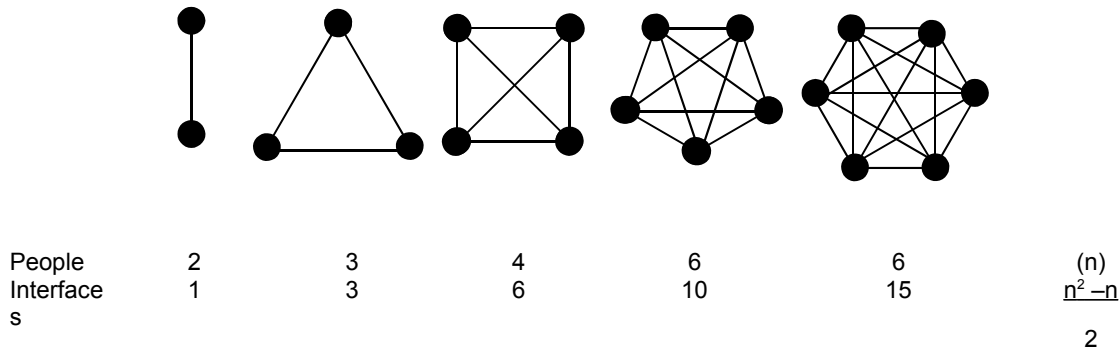
# The Mythical Man Month

In 1975 during the pioneering days of software development a man named Frederick Brooks penned a number of books and articles on the subject. His most famous is “No Silver Bullet”, in which Brooks pointed out that software development could expect no thunderbolt solution to its various problems of quality, cost and complexity other than to adopt rigorous methodology.

Only slightly less famous than “No Silver Bullet” is another Brook’s paper, “The Mythical Man Month”. They are no less valid today than they were then, but they receive a lot less attention.

In “The Mythical Man Month” Brooks argues that adding people to a project doesn’t speed it up. While it is true that more resources *can* speed up the delivery of a software product, the increase in speed is not directly proportional to the amount of resource added. To put it another way, simply adding resources to your project will not ensure earlier delivery.

The main reason for this is the increased complexity of communications which results from adding more people. As each person is added to the project team the complexity of communications goes up exponentially. For each project there is a break-even limit where adding more people will in fact slow down the project.



The diagram above demonstrates the principle graphically. Note that you need not consider each of the ‘nodes’ in the graph an individual person – they could be a group of people or an organisation within the project that has an interface. The more interfaces you add the more complexity you add to communication and the more overhead you add to the project.

If you don’t believe the math, look at it logically. Every additional person brought into a project during the development cycle will need to be trained and briefed on the current status and assigned tasks. As more and more people are added, more of the original team must be devoted to managing the overall structure. This is a truism of all types of management, not just project management.

Yet, while obvious, this mistake is committed time and time again by project managers. The first reaction to any slow-down in the schedule or a threat to the delivery of the project is to throw more people at it. This rarely works in a well-controlled project and *never* in a badly controlled project.

Adding more people to a project requires ‘bandwidth’ to manage them and can distract you from more important goals at hand.

There are a few things to learn from Brook's "Mythical Man Month" :

1. Small autonomous teams are more efficient than large bureaucratic ones, so divide your project up into manageable chunks and let each group work within some kind of defined boundary.
2. If you want to add people to a project, you had better plan carefully how those people are introduced into the team, there will be a lag before they become productive and even be a drain on the productivity of other members of the team. Look for 'flat spots' in the schedule to introduce these people to the team.
3. One of your options in the "scope triangle" has just been reduced! If the scope of your project expands you know there's only a limited benefit in adding more people to the project because of the overheads involved. We're back to those same two options again : ask for more time; or cut functionality!

One particular project I was involved with illustrated to me the truth behind the "mythical man month" more than any other.

I was the consultant test manager representing the client, a major bank. A senior manager in the bank had staked his reputation on the success of this system and now no expense was spared to make the project fly! The developer, one of the world's largest IT service companies, had flown in a design team from overseas since no local talent was available at short notice. They had also flown in a top notch project manager from the other side of the world to see their first project with the bank succeed.

As the project progressed the plans became more and more ambitious and more and more people were added to the project. We started off with one design team and ended up with three, none of which ever received the same brief. The developer started flying in software engineers from a neighbouring country and then flying them home for the weekend. Local staff were diverted to the project to help the interlopers try and meet their deadlines but they were still reporting to their original line managers.

It was chaos. Developers were sitting around waiting for instructions. Graphic designers were busily designing interfaces for screens whose business logic hadn't even been finalised. There were at least three different versions of the specifications floating around and no one knew which one was current.

Our role was to vet the quality of the supplied system for the bank, in effect accepting the system on their behalf. We had a field day! Every release was turned back with major bugs because it hadn't been tested by the developers and was handed over incomplete.

To my knowledge the system was never launched even after our involvement ended. Expenditure on the whole project must have been on the order of tens of millions of dollars and the project ended up on the scrap heap!

You have to know what you are trying to do.

This seems obvious but lack of clarity in the early stages of a project is very common and causes many problems. Many projects start up with vague or ill defined ideas of what they want to achieve. If you hope to deliver a successful project in a finite amount of time you need to determine the final state your product must achieve, you need to set yourself a concrete goal.

If you have an infinite amount of time you could simply try one solution after another until you hit upon the best solution for your problem. This 'inventive' approach to product development can give rise to spectacular and unique solutions but more often than not ends in failure or inadequate results. Also most of us don't operate in environments where we have infinite amounts of time or resources. Most of us operate in an environment where we need to deliver a concrete in solution in a very finite period of time.

In order to do this we need a way to select the best solution from a range of possible approaches. The first and most important step in this process is defining what will actually constitute a success. Then we can evaluate all of the possibilities against our definition of success and find the best fit. Without this we'll be shooting in the dark.

The more disciplined you can be about defining your objectives, the more likely you will be to succeed.

## Scope, Visions and Goals

Scope is a general term to describe everything that your project encompasses, everything that must be achieved for the project to be complete. This would encompass your vision, your goals and your requirements and would be embodied in documents such as a "project proposal" and at a lower level "commercial specifications" and "technical specifications".

The word 'vision' produces shudders in technical and non-technical people the world over. And rightly so, for a vision is normally a collection of meaningless catch phrases and marketing dribble intended to dupe people into thinking that businesses are there for some polite and altruistic reason, rather than to extract every last cent out of their customers. This is not the kind of vision I mean.

When I talk about vision I'm simply saying that you need a single encapsulated idea which defines the aim of your project. Why are you doing the project in the first place? What makes a project a project is the fact that it is a standalone task (or set of tasks) that has an intended outcome. You work on your project, complete it and then move on to the next.

*If you can't state the aim of your project in a single sentence, then it's probably not a project.*

Maybe it's an idea for a business or possibly a way of life but not a project. It might even be a set of projects that need to be divided into single 'efforts'. A project is a defined task with a finite life with a fixed end point and that end is defined by your 'vision'.

Without a single, linking goal all the dependent steps of project planning become difficult to manage. That single vision may be broken up in sub-goals but it provides the link that holds all of the disparate parts of the project together into a single enterprise. It gives your team and stakeholders a sense of purpose and defines the success of your project.

Goals are slightly lower-level and more specific than the vision. Goals should directly support the

overall vision of the project but refine its definition. Typically goals are set out by customers or by a business and define how the success of the project will be achieved. While the vision encompasses the whole project, goals may refer only to the objectives of a particular segment of the project.

Note that the terms scope, vision and goal are largely interchangeable. Different organisations use them in different contexts to refer to much the same concepts. The definitions set out here are the most commonly used versions. Use the version most appropriate to you.

## The Vision as Inspiration

While further steps in the project planning aim to be more and more specific the initial goal should be broad enough to encompass the whole project. The vision must state, succinctly, the ultimate goal for the whole project.

The goal or vision should also be inspiring or, appropriately enough, “visionary”.

Goals like :*“To deliver the cheapest system, in the shortest time, that just about gets the job done”* are unlikely to inspire anyone or motivate a team.

On the other hand a goal like :*“Deliver the best sales and marketing system on the market”* is more likely to inspire personal involvement from team members and stakeholders.

If you are working on your own an inspirational vision can restore your flagging enthusiasm. When the client or your manager calls you up for the nth time and says “Where’s that bit of documentation I asked for ?” and you say to yourself “Why am I doing this again ?” – your answer could be “because I’m writing the best damn <insert name> the world is ever going to see!”.

Visions don’t have to be written down or cast in stone. They don’t even have to be formalised in any particular sense. In large organisations they often are since that’s the kind of the thing large organisations like to do, but the only important thing is that you, your team and your stakeholders know exactly what the vision is and agree on it.

Don’t go overboard, now is not the time to exercise your commercial-buzzword vocabulary. Select language that is natural and easy for you to use and that sounds sincere. The more you believe the vision and the more you use it, the more that other people, including your team and your customers, will come to believe it to and the more chance you will have of succeeding.

One of the most important things you can do is to inspire trust in the people you work with. It is human nature to be sceptical and it is easier for most people to assume that a project will fail rather than assume it will succeed. You don’t have that luxury however!

Everybody from the team to the stakeholders to the man signing the cheque will want reassurance that you know what you are doing. You have to build confidence in yourself and in the project .

Often the vision will be delivered into your hands by your executive sponsor or a client that commissions your project. In discussions with them you will notice that they have a singular way of referring to the project such as *“we want a sales and marketing system that’s going to save us time and money”*. You could do worse than adopt a phrase like that as a vision but consider rewording it to suit your own purposes.

## Goals as a Filter for Requirements

One of the primary purposes of goals is to act as a filter for subsequent requirements.

If a particular requirement cannot be traced back through higher-level goals to the overall project vision then it should be dropped since it will be outside the scope of the project.

For example, if the overall vision for the project was to “*Deliver the best sales and marketing system on the market*” an appropriate sub-goal might be “*to deliver a sales-order processing system for use throughout all international offices*”. An inappropriate sub-goal would be “*to deliver an invoicing system*” since the invoicing system would be part of the financial system and not the sales system.

This process of filtering should be used throughout the life cycle of the project to assess requests for extra functionality and to consider them for inclusion within the project. The use of such filtering techniques gives you an easy method with which to avoid the perils of “scope creep”.

The linkage between low and high level goals can be fundamentally important to your team as well. By linking them together you can track your progress through the objectives of your product. By delivering low level goals you build up to the delivery of high level goals and ultimately the delivery of the completed project. This is known as *traceability*.

Proposal	Requirement Spec.	Technical Spec.	Test Plan
<b>Goals</b>	<b>Requirements</b>	<b>Functions</b>	<b>Tests</b>
G1	R1.1	F1.1	T1.1
G2	R1.2	F1.2	T1.2
G3	R2.1	F1.3	T1.3
G4	R2.2	F1.4	...

## A Decent Proposal

Sometimes referred to as a ‘business case’, the project proposal states the highest level goals in a project. It outlines the overall business goals and vision for the project as decided by the customer or client. It is sometimes drawn up well before the project starts although you may (if you are lucky) also get a hand in its creation.

The basic proposal should contain the vision for the project and the business goals, what your client hopes to achieve at a business level. There may also be a large amount of supporting information in order to qualify or corroborate the stated goals but the goal should be clear. The supporting information might be preliminary forms of the project planning such as budgets, schedules and so forth.

Project proposals are often vital documents because they are what gets signed off when a commercial deal is agreed. As such you need to consider them carefully because they may be define what your are legally committed to delivering.

On the next page is an example based on our hypothetical sales and marketing system. The vision is stated first and after that a list of specific business and technical goals is listed. Each of the specific goals contributes directly to the vision of delivering the sales and marketing system.

## Whizz-Bang Customer Relationship Mangling System

The project should deliver the best Customer, Sales and Marketing system on the market, it should :

- Reduce the time taken to process sales orders by 50% (of manual processing times)
- Provide detailed management reports on a quarterly basis
- Provide detailed market and customer analysis at request
- Link sales directly to marketing initiatives to measure marketing ROI
- Provide detailed client and prospect information to individual account managers
- Completely automate licence renewals via a website
- Provide a zero-footprint client, accessible via the Internet for international offices
- Provide an upgrade path for users of other sales order systems

You will note that this is not long or overly detailed. It provides an adequate framework for moving the project forward without getting bogged down in detail. The goals outlined above will probably be supported by a fair amount of commercial and market research but within the context of the project, the above should be more than adequate to establish the objectives of the project. The level of detail will depend on the size and importance of your project to the organisation.

You should also note that some goals are more specific than others. For example “reduce the time taken to process sales orders by 50%” is a fairly specific, readily testable goal. On the other hand “provide detailed management reports on a quarterly basis” is a little bit vague. What kind of reports? In what format? For whom?

Although more detail is desirable it is probably not necessary at this stage. The broad goals have been laid out and it will be the purpose of subsequent phases (like requirements specification) to define how they will be achieved.

Spend enough time on your project proposal to make sure it is accurate and succinct. It will be the yardstick against which senior management will judge the success of your project. Don't spend so much time on it that you delay the commencement of the project proper.

## **Return on Investment**

One important component of many formal business cases is a 'Return on Investment' or ROI calculation. Simply put an ROI calculation compares the cost of a product with the benefits you expect to achieve. Different projects can then be evaluated on a like-for-like basis and the best use for the money selected.

Obviously to compare costs to benefits they need to be stated in the same 'units' and not surprisingly these are usually “dollars”. This is where the problems start. The cost of a project is usually fairly easy to determine but the benefits can be much harder to quantify. Benefits are usually determined by asking the customer to estimate what benefits, in dollar terms, they hope to achieve through using the product.

This could range from freeing up someone from a manual process (thus saving money) to attracting more customers to a business to simple sales for an off-the-shelf product. All of these are fairly easy to put a dollar figure to but all of them are based on future predictions which can be unreliable. For example how many hours of someone's day will you free up ? How many more customers will this product attract or how many sales do you hope to achieve in the first year ?

Notwithstanding this, if you can confidently estimate one of these values you can then compare

your costs to your benefits. Your return on investment is just the ratio between the costs and the benefits with a positive ratio indicating profit, or a return on your investment. This is normally done over time to determine when the product will reach its 'payback' point.

For example if I decide to make a terrific new product which would help me breed marmosets, I could work out the ROI like this :

1. I estimate that it will take me roughly six weeks to design, develop and debug my product. I will also need a new PC costing about \$2000 and some important marmoset measuring equipment costing about \$500. I also pay myself about \$500 a day, so the cost would therefore be :  $30 \text{ working days} \times \$500 + \$2500 \text{ hardware} = \$17\,500$
2. By surveying my prospective market I determine that there are about 1000 marmoset breeders in my local area. I estimate that I will reach about 10% of them the first year, then as my fame catches on I will sell to another 30% of them and then as competing products come on the market sales will decline back to a steady 10% per year.

Since marmoset breeders aren't rich I decide to charge about \$50 a copy for my software giving me the following benefit calculation :

10% of 1000 is  $100 \times \$50 = \$5000$  in the first year

30% of 1000 is  $300 \times \$50 = \$15000$  in the second year

10% of 1000 is  $100 \times \$50 = \$5000$  in the third and successive years

3. My ROI calculation thus looks like this :

	Year 1	Year 2	Year 3	Year n
<b>Cost</b>	\$17500	\$0	\$0	\$0
<b>Benefit</b>	\$5000	\$15000	\$5000	\$5000
<b>ROI</b>	28%	114%	143%	...

Based on this calculation my product will reach payback in 1 year and 10 months. Not bad for something that only took six weeks to write! Excuse me while I go out and buy some marmosets...

# Requirements

Requirements specification is the process of refining the goals of a project to decide what must be achieved to satisfy the clients.

Sometimes requirements specification takes place before the formal commencement of a project to help identify and select the right for a solution and technology. Often this is known as a feasibility study or project analysis. More typically however some requirements are thrown together (maybe in a proposal) and the real requirements specification occurs only after the project has started.

## Functional Requirements

Functional requirements are the obvious day-to-day requirements end-users and stakeholders will have for the product. They revolve around the functionality that must be present in the project for it to be of use to them.

A functional requirement typically states as “the system X must perform function Y”. This is known as an ‘assertion’. An assertion asserts or affirms a necessary or desirable behaviour for the system or product in the eyes of a stakeholder.

Without clear assertions requirements are nothing more than vague discussions which have a regrettable tendency to clutter up your desk and your mind.

Compare the two following, contrasting, functional requirements:

- *The financial system must produce a detailed customer invoice as per Appendix A.*
- *Producing an invoice for customers is important. Invoices should contain all the pertinent information necessary to enable a customer to supply payment for goods.*

The first is a functional requirement stated as an assertion. It indicates that the *financial system* is responsible for producing a *detailed customer invoice* which contains all the information in *Appendix A*. While it could be more specific, the reader is left in no doubt as to what the financial system *must* do in order to be a successful financial system.

The second could be the introduction for a chapter in an accounting book. Although it states that invoices are important it gives no indication of who or what is responsible for producing them. It then rambles on about what goes in an invoice which everyone already knows anyway. Such a statement does not belong in a requirements specification.

The second ‘requirement’ compounds the problem by looking solid but really being vague and ambiguous. What does *pertinent information* mean? *To enable a customer to supply payment* is superfluous since that's what an invoice is for. The statement, while accurate, contributes nothing towards our understanding of what the system must do to be successful.

Here are some more ‘better’ statements of requirements:

- *A customer account record must contain a unique account reference, a primary contact name, contact details and a list of all sales to the customer within the past sales year*
- *Contact details must consist of a phone number, an address and an optional email address*
- *For each contact up to five separate phone numbers should be catered for*

## Non-Functional Requirements

It is essential to consider the other requirements too, these are called “non-functional requirements” which, to my mind, is a bit of an oxymoron.

Performance	Performance usually covers areas such as responsiveness, throughput and speed of operation. What is the minimum performance that will satisfy your client ?
Usability	How “easy-to-use” will the finished product be ? For example do you cater for disabled or handicapped users ? Generic ease of use should be considered though, more than one product has failed by supplying full functionality with an obscure or convoluted interface.
Reliability	Reliability requirements deal with the <i>continuous availability</i> of the product to users. They should state what availability is necessary and desirable.
Security	In products which deal with confidential or sensitive information, security considerations should be taken into account. Requirements for different levels of access, encryption and protection should be gathered.
Financial	There may be financial considerations which will determine the success or failure of the project. For example a bank or investor might specify certain financial constraints or covenants which must be satisfied during the project.
Legal	There may be legal requirements that must be met due to the environment in which your product will operate. Consult a legal expert for these.
Operational	There may be a number of day-to-day operational issues that need to be considered. Failure to accommodate these will not delay project launch but may limit or halt its uptake by end-users once it has been launched.
Specialist	In every project there are a number of specialist requirements which are dependent upon the nature of the project or the nature of the business. These should be considered separately and explicitly stated within design docs.

## Stakeholders

Stakeholders are an integral part of a project. They are the end-users or clients, the people from whom requirements will be drawn, the people who will influence the design and, ultimately, the people who will reap the benefits of your completed project.

It is extremely important to involve stakeholders in all phases of your project for two reasons: Firstly, experience shows that their involvement in the project significantly increases your chances of success by building in a self-correcting feedback loop; Secondly, involving them in your project builds confidence in your product and will greatly ease its acceptance in your target audience.

There are different types of stakeholders and each type should be handled differently :

Executive	Executive stakeholders are the guys who pay the bills. Typically they are managers or directors who are involved with commercial objectives for the project. They should restrict themselves to commercial considerations and be actively discouraged from being involved in technical design, their experience and skills are vastly different to that of 'typical' end-users.
End-user	These are the guys that are going to use your product. No one knows more about what the product is supposed to do when it hits their desks than they do. No one ! Including you ! You may think you know better but if you don't listen to them you're kidding yourself.
Expert	Sometimes you need input from experts in other fields. People like graphic designers, support reps, sales or sometime lawyers and accountants.

# Requirements capture

Requirements capture is the process of harvesting the raw requirements of your stakeholders and turning them into something useful. It is essentially the interrogation of stakeholders to determine their needs. This can take many forms, with questionnaires or interviews being the most popular. The usual output is a 'requirements specification' document which details which of the stakeholder requirements the project will address and, importantly, which it will not.

The focus in requirements capture must be in gathering of information. Keep your ears open and your mouth shut! Listen carefully to what people want before you start designing your product. Later you can focus on *how* things are to be achieved for now you need to find out *what* must be achieved. (Design decisions involve making assumptions, this can result in 'leading' the stakeholders and delivering a product that you want and not one that they want).

In reality this is difficult to achieve. Technical people complain that stakeholders often “don't know what they want”. This is not true. Stakeholders know exactly what they want – they want less hassle, easier jobs and so on. The problem is that they can't design the system for you, they can't tell you how to achieve what they want. The trick in requirements specification is to take what the stakeholders give you and distil it into something you can use to help you make decisions on how to implement their wishes. One way to think of this is as finding the ideal solution to the stakeholder's current problems.

Requirements capture also needs to be fast. Projects have a tendency to bog down at this stage and to produce reams and reams of documentation, but no useful output. The aim of requirements capture is not to produce an endless tome detailing the answer to every possible question but to provide enough clarity for the project team so that the objectives are clear.

## Questionnaires

Questionnaires are a typical way of gathering requirements from stakeholders. By using a standard set of questions the project team can collect some information on the everyone's needs. While questionnaires are efficient for rapidly gathering a large number of requirements their effectiveness can be limited since it is a one way process. If you don't ask the right questions you don't get the right answers. There is no way to seek clarification or resolve conflict. To resolve this, questionnaires are usually used in conjunction with other methods, such as interviews.

## Interviews

The same questions posed in a questionnaire can be put across a table in an interview. The benefit of the interview is that it allows more exploration of topics and open ended discussion on the requirements for the project. It's a two way process.

Interviews can either be structured as single or group sessions. Particular stakeholders can be interviewed individually or a group session can be used to thrash out ideas amongst a larger number of stakeholders (and hopefully obtain consensus). In a group session you can use a formal structure or a more open style where ideas are thrown, a brainstorming session. The best ideas that survive and can be adopted by the project team as part of the requirements.

The down-side to interviews is that it is time and people intensive. Not only must the interview be set up and conducted but minutes must be taken, distributed and reviewed, and it may be necessary to hold one or more follow-up meetings. Using questionnaires *and* interviews is a common and efficient practice; questionnaires can be distributed beforehand and an overview of the stakeholder's requirements collected. The interviews are then focussed and directed towards the clarification of those requirements.

## User observation

Another method of requirements capture is direct end-user observation or evaluation.

The purpose of user observation is to capture requirements that the end-users may not be consciously aware of. For example individuals using a cheque processing system in a large finance system may conduct a number of manual steps outside of the system that could be automated. Because they don't regard these steps as being part of the system they will not mention them when questioned about the incumbent system. Only by direct observation will the development team become aware of the importance of these steps.

You can either use free form observation or you can take a typical group of end users are set a series of tasks and monitor their processing of these tasks. Notes are made on the way they conduct the tasks, any obstacles they encounter and you could even video tape it (if they agree).

Direct user observation is particularly powerful because, unlike the first two methods of requirements capture, it relies on observed fact and not upon opinions. It is however, the most resource intensive of the three techniques.

## Conflicting Requirements

One or more requirements may contain elements that directly conflict with elements of other requirements. For example, a performance requirement may indicate that a core system must be updated in real time but the size and scope of the system (as defined by other requirements) may preclude this. Updating such a large system may not be possible in real time.

One of the most effective ways of resolving the conflict between requirements is to impose some form of prioritisation on the requirements. This allows the potential for negotiation since it provides a basis for assessing conflicting requirements. For example if, from the previous example, the requirement for real time updates was rated at a much higher priority than the inclusion of full customer data then a compromise could be reached. The main 'online' database could contain only the barest essential of customer details (allowing real time updating) and a separate 'archive' database could be established which contained customer histories.

Requirements are often heavily interlocked with other requirements and much of the time it seems your stakeholders have diametrically opposed points of view and cannot come to terms. If you pick apart any set of requirement you can come to some sort of compromise with both parties and achieve consensus. This can be a difficult and even emotional process.

The very best way I have seen to resolve conflicting requirements works like this :

1. The stakeholders draw up a list of their requirements
2. The list is organised in strict priority order with the most important at the top, down to the least important at the bottom.
3. The project team then looks at the schedule and draws a line through the list based upon what they believe they can deliver with the time/money available
4. The stakeholders assess the development position and can then re-prioritise their requirements if required or negotiate over the nature and importance of requirements
5. Once consensus has been achieved both sides sign-off and work starts

This is not a simple or easy way to achieve consensus however. Even the basic ordering of the list of requirements is no mean feat in a project of any size.

# Documenting Requirements

You need a way of documenting your requirements for your project team. Stakeholders are also often asked to sign off their requirements as a confirmation of what they desire. Often this is where money starts to change hands.

Documenting requirements is much more than just the process of writing down the requirements as the user sees them. The requirements specification is an essential link in the total design of the whole project and attempts to give meaning to the overall goals of the project.

Whatever form of requirements documentation is used it should cover not only *what* decisions have been made but also *why* they have been made. Understanding the reasoning that was used to arrive at a decision is critical in avoiding repetition. For example, if a particular feature has been excluded because it is simply not feasible that fact needs to be recorded. If it is not, then the project risks wasted work and repetition when a stakeholder requests the feature be reinstated during development or testing.

Documenting the decision process is also useful from a stakeholder's point of view because it allows the stakeholders to better understand what to expect from the final product. A basic statement of requirements without any underlying discussion can be difficult for a layman or end-user to understand.

## SMART requirements

One useful acronym for defining requirements is SMART :

<b><u>Specific</u></b>	A goal or requirement must be specific. It should be worded in definite terms that do not offer any ambiguity in interpretation. It should also be concise and avoid extraneous information.
<b><u>Measurable</u></b>	A requirement must have a measurable outcome, otherwise you will not be able to determine when you have delivered it.
<b><u>Achievable</u></b>	A requirement or task should be achievable, there is no point in setting requirements that cannot realistically be achieved.
<b><u>Relevant</u></b>	Requirements specifications often contain more information than is strictly necessary which complicates documentation. Be concise and coherent.
<b><u>Testable</u></b>	In order to be of value requirements must be testable. You must be able to prove that the requirement has been satisfied. Requirements which are not testable can leave the project in limbo with no proof of delivery.

## The Language and Layout of Requirements Specifications

Requirements specs. often have a lot of information in them. Mainly because they have multiple audience. They are used by the project team to deliver the product, and they are used by stakeholders to verify what is being done. While I advocate including contextual information to illustrate why a particular decision was taken there needs to be some way of easily separating the discussion from the “assertions”.

So I apply the following rules of thumb:

- For each requirement there should be an “assertion”; in essence, a decision
- Where assertions are documented they should consist of a single sentence which states what a product “must” or “should” do.

- “Must” indicates a necessary requirement and “should” indicates a “nice-to-have”.
- There must be simple (visual) way to distinguish assertion from discussion

Below is an example with some discussion and the assertion highlighted with *italics*:

**1.1 Usability** – usability of the system is seen as very important to adoption of the system. The client raised some concerns about the timescales required to implement usability testing but the project team as a whole supported extending the time line for the development phase to include usability testing..

*The system must be simple and easy to use and must follow the standard UI style as laid out in the company design handbook.*

In this case the discussion is preserved for future reference but the requirement stands out distinctly from the body of the text. A project member reading the specification can skip through it and pull out the requirements quickly and easily. A manager reading through the document can do the same but also can review the context of the decision to understand how it came about.

Sometimes classification of requirements is made using the MoSCoW rules:

**M**ust-haves are fundamental to the project’s success

**S**hould-haves are important, but the project’s success does not rely on these

**C**ould-haves can easily be left out without impacting on the project

**W**on’t-have-this-time-round can be left out this time and done at a later date

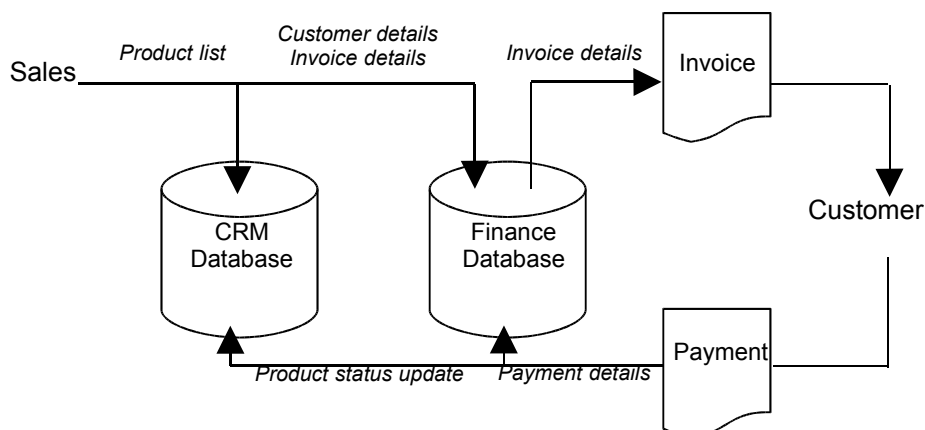
I feel that the distinction between “should have” and “could have” is never clear and is usually the subject of much debate between client and project manager so I normally omit “could have”. Every requirement then either becomes necessary (“must have”) or optional (“should have”). Stick to prioritising them.

## Diagrammatic Methods

While the emphasis in this chapter has been on requirements specification in the form of text, other more graphical methods are available. As the saying goes, a picture is worth a thousand words, and this is both a blessing and a curse. While representing information with a picture or diagram can be extremely informative it can also be extremely confusing. Diagrams can often imply requirements without actually stating them and leave details open to interpretation.

For this reason I see graphical methods as supporting standard textual methods of description. They should be used wherever appropriate, where the graphical nature of a representation will more closely represent the nature of the requirement. If you find it difficult to explain in words the nature of a particular structure or process then by all means insert an appropriate diagram.

The emphasis must be on concise, accurate representations, so use whatever combination of graphical and textual elements seems right to you.



## A Sample Requirements Specification

The most common method is to break down the requirements in an outline fashion as used in a document or manual.

For example:

1. **Current product status** – all parties highlighted the need for a clear and public indicator of the current status of a product
2. **Dates** - dates for each major milestone were also recognised as necessary. Although some of these dates will remain in the public domain others will be available only to “private” users. Private users will have the ability to publicise dates as they see fit.  
The dates specified are:
  - 2.1 **Development sign off**
  - 2.2 **Testing sign off...**

Even more structure can be put into the document by splitting up requirements categories :

For example:

1. **Functional Requirements**
  - 1.1. **Product list** – *the system should produce a list of products available or under development*
  - 1.2. **Current product status** – all parties highlighted the need for a clear and public indicator of the current status of a product. *There should be a simple flag which indicates at-a-glance whether the product is ready for release.*
  - 1.3. **Dates** – *for each product, dates for each major milestone must be shown. Although some of these dates will remain in the public domain others will be available only to “private” users. Private users should have the ability to publicise dates as they see fit.*  
The relevant dates are listed below:
    - 1.3.1. **Design sign off**
    - 1.3.2. **Development sign off**
    - 1.3.3. **Testing sign off**
    - 1.3.4. **etc...**
2. **Non-Functional Requirements**
  - 2.1. **Performance** – *the system must be updated daily and information available to all international users within 1 min of the information being posted by head office.*
  - 2.2. **Usability** – *usability of the system was seen as very important to adoption of the system. The system must be simple and easy to use and must follow the standard UI style as laid out in the company design handbook.*
  - 2.3. **Security** – *access to schedule information must be controlled on a per-user basis. Access to the information should not be available to any external customers or companies.*

It is best to be as specific as possible but remember the 10th commandment and be flexible. If your project doesn't warrant this level of detail then don't include it; or you will spend all your time writing documentation. Find a happy medium between detail and effort that suits you and your organisation's needs.

# Traceability

One of the most time-consuming overheads is that of managing traceability.

Given a reasonably complex project with hundreds or perhaps thousands of stakeholder requirements how do you trace the fulfilment of a single requirement from specification, through design, development and ultimately to testing and launch?

How do you prove at test or implementation time that a requirement has been satisfied? How do you track the progress of delivery on a particular requirement during development? And how do you ensure that your design incorporates all of the particular requirements as specified in earlier phases?

The simplest method has already been outlined in the examples above. By using a common format for both the requirements specification and the technical specification it is relatively easy to track a requirement through to a design element. If the test plan is also based on the same format then the traceability can be extended to the testing phase and it can be shown in test reports — which test validates which design element and, consequently, which requirement.

Proposal	Requirement Spec.	Technical Spec.	Test Plan
<b>Goals</b>	<b>Requirements</b>	<b>Functions</b>	<b>Tests</b>
G1	R1.1	F1.1	T1.1
G2	R1.2	F1.2	T1.2
G3	R2.1	F1.3	T1.3
G4	R2.2	F1.4	...

However, this alone is often not enough. With large-scale projects the sheer number of requirements overwhelm this kind of formatting. It is also possible that a single requirement may be fulfilled by multiple elements in the design or that a single element in the design satisfies multiple requirements. This makes tracking by simple reference number difficult.

It is possible to cross-reference requirements versus design decisions in a separate database or spreadsheet. This, however, incurs a large maintenance overhead and can be very difficult to keep in synch with the parent documents. The best solution remains an appropriate structure. While formatting and structure may not provide one-to-one traceability for every requirement, the use of an appropriate structure will minimise confusion and may eliminate the need for more complicated solutions.

At the extreme end of the scale you may wish to build or buy an integrated system to track requirements for you. There are off-the-shelf software tools available which use large databases to track individual requirements as database elements. These are then linked to similar items in specification and testing databases to provide the appropriate traceability. The benefit of a system such as this is that it can automatically produce reports which highlight problems with traceability.

## The Purpose of a Project Plan

The purpose of a project plan is to maintain control of a project.

As a complicated process, a project always threatens to exceed the limit of your control. Some people are better than others at controlling complex problems, but all of us reach our limits at some stage. To maintain control you need help in the form of tools, your best tool is your plan.

The project plan controls the project by:

- Breaking a complex process down into a number of simpler components
- Providing visibility for obscure or ambiguous tasks in the project
- Providing a single point of reference for everyone
- Enforcing scrutiny of the sequence and nature of events
- Providing a baseline against which the actual execution of the project can be compared
- Anticipating likely events and providing pre-planned means of avoiding them

A project plan must be as accurate, complete and as specific as possible. How accurate, complete and specific of course depends upon how much time and resource you have available.

## The Elements of a Project Plan

Every project planning methodology has its own specific taxonomy and names for its parts. But in a very broad sense the minimum elements a project plan *must* specify are:

*What is to be done* – what is desired of the project and what it must deliver to succeed. This is a scope document at a high level and requirements specs. at lower levels

*When it needs to be done by* – the deadlines by which the objectives must be met

*Who is to do it* – The people, sometimes unkindly labelled “resources”, or the team who are to deliver those objectives. This also usually implies costs since in most projects the application of costs implies the use of skilled labour

*How it is to be achieved* – This is normally in documents such as a technical specification

Note that you do *not* need to complete all of these prior to starting your project. Typically one draft of the proposal, schedule and budget are completed before your project commences. Each of the other documents will be completed at some point through the lifecycle. Nor should any of these documents be regarded as static or inert manuscripts. Each is a living breathing expression of the project at a particular point in time and they should evolve as your project evolves.

Finally, none of these documents has a obligatory size, format or length. Although I suggest various forms and styles they are examples rather than as strict templates. Remember, these are tools not doctrine! If it is easier and more efficient to scribble your project schedule on the back of a cocktail napkin then do it! It is the objective that counts, not the form. The only right form is the one that works for you.

# The Fine Art of Scheduling

Why the fine “art” of scheduling?

If it were a science then every project would be delivered on time!

This sadly does not seem to be the case. In fact, overruns have become so common that people have lost faith in project deadlines and view them with a great deal of cynicism.

In truth the art of scheduling is based on experience and the more experience you have, the more accurate your schedule will be. However, you can still produce an accurate schedule by following some simple rules.

## Principles of Scheduling

**Rule #1** - Never,, give off-the-cuff or unconsidered responses  
(don't commit to something you can't deliver).

Scheduling is one part prediction and one part expectation management. If you are pressured into picking a date “on-the-fly” at a random meeting you can bet that the date will not only be wrong, it will come back to haunt you. A considered response when you have had time to evaluate all the factors is much better. A date picked out of the air is good to no-one, least of all yourself.

More times than I can remember I have sat and watched an inexperienced project manager in a meeting with clients or senior management blurt out a date or like “four weeks from today”. Not one of them to my knowledge ever delivered, it was just a response to pressure, a desire to please.

If management pushes you, give a realistic answer if you have one or ask for more time to formulate a proper estimate – remember it's your project and you will only be undermining your own success by giving an ‘off the cuff’ response.

**Rule #2** - Eliminate uncertainty wherever you can.

The more specific you can be in your project planning, the more accurate your schedule will be. If you leave functionality or other items unspecified in your plan, then you will, at best, only be able to approximate them in the schedule. Don't go overboard, though, there is a balance. If you are spending time adding detail to tasks which will have no impact on the project delivery date, then you are probably wasting your time.

**Rule #3** - Build in plenty of contingency to cope with variation.

No matter how well specified your project and how accurate your schedule, there will be the inevitable random influences that will wreck your carefully crafted schedule. People get sick, equipment fails and external factors join together in a conspiracy to see that you miss your target date. In order to buy yourself some insurance you should build in an adequate amount of contingency, so that you can cope with unexpected delays.

You should also spread contingency throughout your project timeline and not just place it at the end. If you only have one pool of contingency allocated to the end of your project you are leaving yourself with a large slice of uncertainty. By breaking it up and spreading it throughout your project you allow yourself more options and are able to control the project more closely. You can also “buy back” time when you return unused contingency to the project.

Contingency really refers to the Vth commandment, or “promise low / deliver high”. It is better to be pessimistic in your estimations and then surprise people with a better than expected performance than it is to cut your estimation to the bone and blow your schedule at the first hiccup. The old adage “think of a number and then double it” has some validity when it comes to project scheduling.

#### Rule #4 - Pick the right level of granularity

When drawing up your schedule it is important to pick the right level of detail. If you are going to require daily updates from your team then it makes sense to break into day-by-day chunks. That way everybody has the same understanding of what must be achieved by when.

On the other hand if your project has large portions of time devoted to similar activities, testing for example, then it may be better to simply block-schedule one or two months of testing. Maybe you can leave the details up to your team, it all depends on the level of control you want.

In most projects I've dealt with my optimum level of granularity is a week. This means that tasks are scheduled on the basis of the number of weeks they take. Week-by-week is much more comfortable for most people since finishing a task by the end of the week seems more natural than finishing it on a Monday or Tuesday.

Day by day scheduling can provoke more overhead than you really need. If a task is scheduled to be completed on Wednesday but due to difficulties it cannot be completed, it is unlikely that it will be finished on the Thursday, even if a team member predicts it to be so. It is more likely it will overrun by a couple of days and finish sometime on Friday, meaning that subsequent tasks can't take place until the next week. If I schedule day-by-day then I spend all of my time updating the schedule and not managing the project. On the other hand if I schedule week-by-week it is much easier to cope with such small variations. If something scheduled for “the week beginning Monday the 21st” is delayed by one, two or even three days, then subsequent tasks can either be moved comfortably or may not even be affected at all (depending on my level of contingency).

The only exception to this is where I need to force the pace of a project. I do this by imposing tighter deadlines, to the day or even down to the hour, for completion of tasks. A higher level of control however implies a higher level of attention and if I do this, I know it has implications for my own work-load as well. On a finer grade of schedule I will need to pay closer attention to individual tasks to ensure their completion.

#### Rule #5 - Schedule for the unexpected

Project management is the art of handling the unknown. Often events and circumstances you could not have foreseen will interrupt the flow of your project. It's your job to take them all in your stride. Schedule for the most likely delays and cope with them should they arise. If experience or instinct tells you that a certain type of task will overrun, then anticipate it, pad it with some contingency and make sure you have adequate resources on hand when it comes up.

A good way to cope with this is to implement a bit of impromptu risk management (see Risk Management). By anticipating likely risks and prioritising them you will be better able to deal with the unexpected! It also makes a lot of sense to let someone else help assess the risk to your project.

## The Format of Project Schedules

### Milestones

A schedule denotes a sequences of events in time. In a project, “milestones” usually denote the completion of significant portions of the project or the transition from one phase to another. At each milestone there will be “deliverables” which must be completed to move on to the next phase.

As previously discussed, a schedule can include any level of granularity that is reasonably practical. In fact a complex project may involve different levels of scheduling. You might work on a weekly schedule with high level milestones while other team members will have smaller scale schedules with finer grade milestones.

Typically, more detail is involved in projects where there are a number of parallel activities taking place. Where more than one person is working on a project at one time it is obviously possible for these individuals to take on separate areas of responsibility and complete them concurrently. In this case the concept of a critical path comes into play and milestones represent those critical events where ‘branches’ rejoin the critical path of a project.

### A Simple Example

In the following example a simple development project is shown which lists all of its major milestones. The usual cycle of design-develop-evaluate has been concatenated into a single “production phase” which starts on the 22nd of January and is finishes on the 1st of March.

Milestone	Completion
Project starts	01-Jan
Complete scope and plan	10-Jan
Specify requirements	21-Jan
Production	1-Mar
Implementation	8-Mar
Review	15-Mar

This example is extremely simple and anything more than a one-person project would be unlikely to succeed using this method. The simplicity makes it easy to understand but as a control tool it is not very useful. The extreme high-level of the schedule means that the only indication a project team will have that a project is not on track is when it misses a major milestone. This could be embarrassing, if not terminal, for the project.

Also there is no contingency scheduled between tasks which could lead to complications or failure of the project to deliver on time. By assuming that everything will run to plan the project are leaving themselves very little room to manoeuvre should anything go wrong.

## A More Complicated Example

In the following, more realistic example, a project is broken down into a number of phases and the start and finish of each phase is recorded separately. Columns have also been added to include the duration of each phase and the deliverables to be completed at the end of that phase.

This new project schedule includes three *scheduled* iterations of the production cycle of design-develop-evaluate. In reality the three cycles may be many more as prototypes are rapidly produced during the cycle but the schedule imposes a *minimum* of three cycles. The deliverables for these production cycles are an “alpha”, “beta” and “final” version of the system.

Phase	Start	Finish	Duration	Deliverable
Scope and plan	01-Jan	10-Jan	7 days	Project proposal
Specify requirements	13-Jan	24-Jan	10 days	Requirements spec.
Production phase 1	3-Feb	14-Feb	10 days	Alpha
Production phase 2	24-Feb	7-Mar	10 days	Beta
Production phase 3	17-Mar	28-Mar	10 days	Final candidate
Acceptance testing	7-Apr	11-Apr	5 days	Release system
Implementation	21-Apr	25-Apr	5 days	-
Launch	28-Apr	28-Apr	0 days	-
TOTAL			67 days	

The first thing to note is that the “Duration” column does not match the number of *calendar* days between the scheduled start and finish dates. This is because the duration column indicates the number of *working* days between the two dates. For example in 2003, while there are 10 calendar days from the 1st of January to the 10th of January (inclusive) there are only 7 working days due to the fact that the 1st is a public holiday, New Year’s Day, and the 5th and the 6th are a weekend .

The next important point to note is that the phases of the project no longer occur contiguously. That is, there are distinct gaps between some phases of the project. For example between the specification of requirements (which concludes on the 24th of January) and the commencement of the first production phase (on the 3rd of Feb) there are 5 working days unaccounted for. These days are held in reserve by the project as contingency.

In the likely event of things not going to plan the project can use this contingency to maintain his schedule and his original delivery date. In the unlikely event that things go exactly to plan the project can simply scrap the contingency and move the schedule forward to the next phase (with subsequent reduction in delivery time). Either way, everyone is happy!

How much contingency to include is a matter of experience. While it can inflate delivery times considerably it can also save the project from damaging and costly failure. In the above example contingency amounts to nearly 50% of the overall production time and possibly represents an excessive amount of caution on the part of the project. As a rule of thumb 10-20% is normal, although more is not uncommon. Take as much as you think your project can live with.

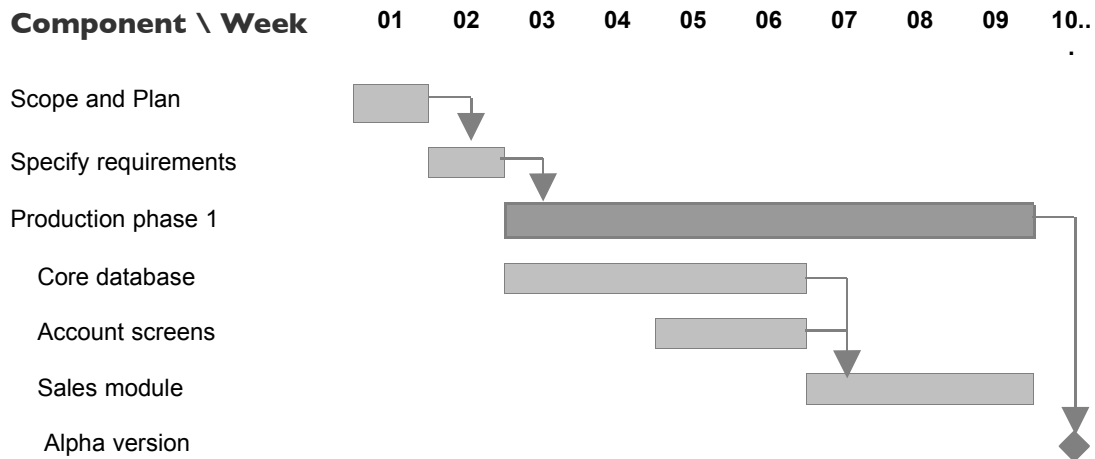
One of the problems with contingency is that as soon as people become aware of it they naturally want to use it. Designers, developers and testers will all want to take advantage of the maximum available time to ensure they deliver the best job, even though your schedule may not permit it.

You can’t pretend the contingency doesn’t exist; all of your team will know about it anyway. You have to appeal to their professionalism to avoid intruding on time which is scheduled for genuine emergencies.

One effective but often neglected method of ensuring that contingency remains untouched is to offer success bonuses for on-time delivery or, more properly, introduce financial penalties for late delivery. This often proves unpopular but effective!

## Gantt charts

One useful tool available to project managers is a Gantt chart. This is simply a visual representation of a schedule. In a Gantt chart, time is represented along a horizontal axis and tasks listed down the left-hand side. The duration of a task is then represented in the body of the graph by a horizontal bar. Milestones are usually represented by single points or diamonds. Dependencies between tasks are also shown as linked arrows. An arrow indicates the necessary order of completion for each task and therefore the progress of the project (including the critical path).



In the above diagram you can also note that “Production Phase I” has been broken down into three sub phases. Delivery of the “core database”, “account screens” and “sales module” have all been estimated separately to come up with the overall estimate for the first production phase and delivery of the Alpha version milestone. Note that either different people are working on the “core database” and “account screens” (since they are happening in parallel) or someone is working overtime!

While this visualisation can be a great help in organising a schedule it must be remembered that the Gantt chart is only a tool. The project manager who constructs the most elaborate and pleasing Gantt chart is not necessarily the one who will complete his project on time. Far too often project managers are lured into the intricacies of artefacts like Gantt charts and spend more time constructing them than managing the project. A Gantt chart is a tool to help you design, monitor or communicate a schedule. If it does none of these — then it is simply a useless but pretty toy.

Be particularly wary of project managers who produce multi-coloured or multi-layered Gantt charts. Also be wary of anyone who produces a Gantt chart with spaghetti like dependencies. More often than not these works of genius are an easy way for the project manager to distract or bamboozle the unwary.

While management are sitting there puzzling away at a labyrinthine Gantt chart the project manager will be high tailing it to god knows where with the project.

To my knowledge, a Gantt chart has never delivered a project by itself!

# Costing and Budgeting

If scheduling is an art then costing could be considered a black art. Some projects are relatively straightforward to cost but most are not. Even simple figures like the cost per man/hour of labour can be incredibly difficult to calculate and in most cases approximations are used.

Accounting, costing and budgeting are extensive topics in themselves and I will not attempt to cover all them here. Instead we will focus on the specific application of costing and budgeting to projects and attempt to give you a grounding in the necessary terminology and principles.

Some fundamental principles to keep in mind are derived from standing accounting practice:

- The concept of 'prudence' – you should be pessimistic in your accounts (“anticipate no profit and provide for all possible losses”). Provide yourself with a margin for error and not just show the best possible financial position. It's the old maxim: promise low-deliver / high once again
- The 'accruals' concept- revenue and costs are accrued or matched with one another and are attributed to the same point in the schedule. For example if the costs of hardware are in your budget at the point where you pay the invoice, then ALL the costs for hardware should be “accrued” when the invoice is received.
- The ‘consistency’ concept. This is similar to accruals but it emphasises consistency over different periods. If you change the basis on which you count certain costs you either need to revise all previous finance accounts in line with this or annotate the change appropriately so people can make comparisons on a like-for-like basis.

Note that the principles are listed in order of precedence. If the principle of consistency comes into conflict with the principle of prudence, the principle of prudence is given priority.

## Costing

At a basic level the process of costing is reasonably simple. You draw up a list of all your possible expenditure and put a numerical value against each item; the total therefore represents the tangible cost of your project. You may also however need to consider “intangible” items.

### Tangible costs

- Capital Expenditure – any large asset of the project which is purchased outright. This usually includes plant, hardware, software and sometimes buildings although these can be accounted for in a number of ways.
- Lease costs – some assets are not purchased outright but are leased to spread the cost over the life of the project. These should be accounted for separately to capital expenditure since the project or company does not own these assets.
- Staff costs – all costs for staff must be accounted for and this includes (but is not limited to): salary and pension (superannuation) costs; insurance costs; recruitment costs; anything which can be tied directly to employing, training and retaining staff.
- Professional services – all large-scale projects require the input of one or more professional groups such as lawyers or accountants. These are normally accounted for separately since a close watch needs to be kept upon expenditure in this area. Without scrutiny the costs of a consultant engineer, accountant or lawyer can quickly dwarf other costs.
- Supplies and consumables – regular expenditure on supplies is often best covered by a single item in your budget under which these figures are accrued. They are related to overhead below.

- One-off costs – one-off costs apply to expenditure which is not related to any of the above categories but occurs on an irregular basis. Staff training might be an example. While it might be appropriate to list this under staff costs you might wish to track it independently as an irregular cost. The choice is yours but the principles of prudence and consistency apply.
- Overheads – sometime called indirect costs, these are costs which are not directly attributable to any of the above categories but never-the-less impact upon your budget. For example it may not be appropriate to reflect the phone bill for your project in staff costs, yet this still has to be paid and accounted for. Costing for overheads is usually done as a rough percentage of one of the other factors such as “staff costs”.

### Intangible costs

It has become fashionable to account for “intangible” assets on the balance sheets of companies and possibly also projects. The argument goes like this: some contributions to a project are extremely valuable but cannot necessarily have a tangible value associated with them. Should you then account for them in the budget or costing? The “prudence” principle says yes but practicality says “no”. If you are delving this murky area of accountancy you should seek professional help and advice.

Typical things you might place in the budget under intangibles are “goodwill” and “intellectual property”. Personnel-related figures are a frequent source of intangible assets and so you might find things like “management team”, “relationships” and “contacts” on an intangibles balance sheet.

### **Budgeting**

Once you have costed your project you can then prepare an appropriate budget to secure the requisite funds and plan your cash flow over the life of the project. An accurate cost model will of course entail a fairly detailed design or at the very least requirement specification so that you can determine your scope of work. This is normally completed well into the design phase of the project.

The sad truth of the matter however is that more often than not you are required to prepare some sort of indicative budget before approval of the project — and you are often held to your original estimates. You must be extremely careful with initial estimates and always follow the “promise low / deliver high” commandment.

Costing and budgeting follow the iterative life cycle as do other tasks within the project. As you refine your design, so you will need to refine the costing which is based upon it.

As in scheduling, you need to build in adequate contingency (reserves) to account for unexpected expenditure. For example, if due to a failure in the critical path a task is delayed and a milestone (like software purchase) falls due in the month *after* it was scheduled. This can wreck your carefully planned cash flow. But if you have carefully budgeted your project then variations should be relatively easy to spot and cope with as they arise.

Just as in scheduling you should have regular budget reviews which examine the state of your finances and your expenditure to date and adjust the planned budget accordingly.

# Risk Management

Risk management is one of the essential phrases in project management these days so I'm going to go out on a bit of a limb when I suggest that it's actually not that important. Understanding the concepts of risk management will be useful for you but I have rarely, if ever, seen it used properly.

The basic concept of risk management is a "chicken-little" assessment of the project which identifies anything significant that could possibly go wrong with the project. A risk management plan will then seek to mitigate or eliminate those risks, hopefully avoiding their consequences. Normally project managers and business managers use it as a kind of 'code' to describe when the project has stuffed up. "We're going to mitigate this risk!" they say as the deadline slips away.

One of the best uses of risk management is to have a "risk" item in your regular project meetings to highlight to everyone the risks that the project currently faces.

## Risk Management Officer

One way for larger-scale projects to control their risk is to appoint a risk management officer. The risk management officer's responsibility is to identify all the risks to a project and to prioritise and present them to the project team for resolution. In smaller-scale projects, risk management is often intrinsic to the role of the project manager and is not considered separately. The principles outlined in this section can still be utilised by smaller project teams, however. Rather than appointing a separate risk management officer a discussion of likely risks can simply be added to whatever regular team meetings occur. A quick discussion amongst different representatives of the team can often highlight some interesting items which have been concealed from the rest of the project team. The formalised process of profiling and resolving risks can then be applied in an informal manner to plan for potential threats to the project.

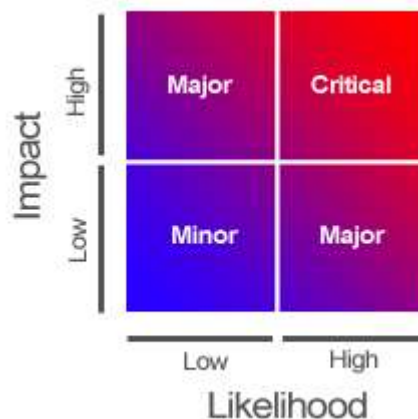
## Risk Profiles

Risk to a project can be measured on two major axes: *likelihood* of failure and *impact* of failure.

The more likely a problem is to occur, the more risk it poses the project. Even fairly minor problems or issues can become a threat to the project if they occur so frequently that they can't be avoided. Similarly, the impact or consequences of a problem are also important. Some problems can stop a project in its tracks all by themselves.

Many systems exist for categorising risks into different categories but the one presented here is fairly simple. In this system each risk item is qualified on two scales: *likelihood* and *impact*. Each scale is divided into two simple categories of "low" or "high" and risks are rated according to each scale.

A "critical" issue represents one that will stop the project in its tracks (known as a "show stopper") and must be dealt with immediately. "Major" risks represent a significant threat to the project because of their frequency or because of the seriousness of their impact; these threats usually have to be dealt with as soon as possible. The third category of risks are "minor" risks which are neither likely nor particularly serious and can be left until others have been dealt with. Minor risks however have an annoying habit of turning into major ones when your back is turned.



## Resolution of risks

Once you have profiled your risk they can be ranked into an ordered list representing the various threats to the project to be dealt with. The more significant can then be examined and assigned an action by the project team.

Typical actions are:

Research	The risk is not yet fully understood. Its impact or likelihood of occurrence may be unclear or the context in which it may occur could seem unreasonable. Further research by members of the project team is warranted.
Accept	The risk is unavoidable and must be accepted as-is. This category of risks become extremely important to a project since they cannot be resolved but still represent a threat to completion. Anticipation therefore become the key to dealing with this category of risk.
Reduce	The risk as it stands is unacceptable. The project team must act to reduce the risk and to establish contingency plans should the risk occur. The risk will have to be reviewed in future to define the threat it poses.
Eliminate	The risk is unacceptable under any circumstances and must be eliminated as a possibility. The project team must put in place processes and procedures not only to ensure the immediate threat is eliminated but that it does not re-occur in the future.

## Recording risks

The simplest way of recording risks is with a table or spreadsheet that lists the risks and their priorities. This can then be regularly reviewed by the project team and action taken appropriately to mitigate or eliminate those risks.

Below is an example of a sample risk table for a project:

Description	Frequency/Impact	Severity	Timeframe	Action
Short-term absence of key team members	High/Low	Major	Project	Accept
Long-term absence of key team members	Low/High	Major	Project	Reduce
Supplier X does not deliver product Y	Low/Low	Minor	Phase I	Accept
Incorrect scheduling	High/High	Critical	Project	Reduce
...	...	...	...	...

In the above table, failure by suppliers to deliver some components has been rated as a minor risk. This sort of judgement can only be made on the basis of experience and within the context of the current project. If the supplier is well known and trusted, then the likelihood of them delivering late is likely to be low and hence the risk can be classified as minor.

Labelling scheduling as a critical area of project risk is also an outcome of experience. If previous projects of a similar nature have run-over due to scheduling problems then it is highly likely that this project will suffer a similar problem. Here, too, you can see the benefits of having a separate risk-management officer since it is unlikely a project manager, however honest, would rate his own scheduling abilities as "high" risk.

# Change Management

A project of any significant length will necessarily deviate from its original plan in response to circumstances. This is fine as long as the change is understood. If the change is not managed but is happens at a whim, it is no longer a project, it's anarchy!

Change management is a way of assessing the implications of potential changes and managing the impact on your project. For example a change in client requirements might mean a minor fix or it might mean a complete re-write of the design. Change management gives you a process to evaluate this and introduce the change in a controlled fashion.

Since change is inevitable you need a fluid way to handle the inputs to your project. It is important that the inputs to your project, your requirements and your design, are able to handle change and evolve over time. If your inputs are static, unchangeable documents then you are going to be hamstrung by their inability to keep pace with changing circumstances in your project.

The most important aspect of change control is to actually be able to know what has changed.

On one product I worked with 30 or more programmers and there was no real change control. Every day the programmers would check in changes to the software and every night we used to run mammoth automated tests, processing 1.5 million data files and producing about 500 lines of test reports.

One day we'd come in and find that our results had improved 10-20% overnight. The next day we'd come back to find the product had crashed after the first thirty minutes and was unusable. The problem was we didn't know who or what was responsible, there was no change control.

Eventually we implemented a system and we were able to make some solid progress towards our goals.

## Change Management Process

The basis of change management is to have a clear process which everyone understands. It need not be bureaucratic or cumbersome but it should be applied universally and without fear of favour. The basic elements of a change process are :

- What is under change control and what is excluded ?
- How are changes requested ?
- Who has the authority to approve or reject changes ?
- How are decisions upon approval or rejection are documented and disseminated ?
- How changes are implemented and their implementation recorded ?

The process should be widely understood and accepted and should be effective without being bureaucratic or prescriptive. It is important for the project team to be seen to be responsive to client needs and nothing can hurt this more than an overly-officious change control process. Change is inevitable in of a project and while you need to control it you do not want to stifle it.

A typical process might be as minimal as the following:

1. Once a project document has been signed-off by stakeholders, a change to it requires a mandatory *change request* to be logged via email. The request will include the nature of the change, the reason for the change and an assessment of the urgency of the change.
2. A "change control board" consisting of the development manager, test lead and a product manager will assess the issue and approve or reject each request for change. Should more information be required a member of the change control board will be assigned to research the request and report back.

3. No change request should be outstanding for more than a week.
4. Important or urgent requests should be escalated through the nearest member of the change control board.
5. Each change which is accepted will be discussed at the weekly development meeting and a course of action decided by the group. Members of the development team will then be assigned to implement changes in their respective areas of responsibility.

If you have a flexible change request process team members can be encouraged to use it to seek additional information or clarification where they feel it would be useful to communicate issues to the whole project team.

## **Tracking Change**

To make change management easy you need a simple method of tracking, evaluating and recording changes. This can be a simple database or log but in large projects it has evolved into an customised information system in its own right.

As such the system needs to be able to handle:

- Logging requests for changes against products and documentation
- Recording and managing the priority of a particular change
- Logging the decision of a change management authority
- Recording the method and implementation of change
- Tracking implemented changes against a particular version of the product or document

The more structured a system the more secure the change control process, but obviously the more overhead. A balance must be struck between the enforcement of proper procedure and responsiveness of the system.

Change management systems are useful for managing everyone's expectations too. Often decisions are requested by stakeholders or clients and if proper consultation is not entered into they can sometimes assume they will automatically be included (just because they asked for it).

If the volume of change requests is particularly high (as it often is) communicating what's in and what's out manually can be difficult. A simple, well understood change management system can often be directly used by stakeholders to log, track and review changes and their approval. This is particularly true for projects that span disparate geographical locations where meetings may not be possible.

In many projects the change management system can be linked to (or is part of) a defect tracking system. Since resolution of a defect is, in effect, a request for change both can often be handled by the same system. The change and defect tracking system can also be linked with version control software to form what is commonly known as a Software Configuration Management (SCM) system. This integrated system directly references changes in the software against specific versions of the product or system as contained in its version control system. The direct one-to-one reference cuts down on management overhead and maintenance and allows the enforcement of strict security on allowable changes.

## Staying on track

The single biggest problem for a project during the execution phase of the project is staying on track. Despite all the best planning, having the best team and anticipating all the possible pitfalls projects seem to have a knack of developing unforeseen problems.

### The Myth of Completion

There is a common falsehood about tasks which is promoted by project tools such as Microsoft Project. The myth is that tasks can be partially complete, i.e. a task can be 10% or 20% done.

If a task is thought of as a goal then the lie becomes obvious – either you have achieved your goal or you have not. It's a black-and-white, binary proposition. If your goal is a vague and imprecise statement of intent, like “write some instructional documentation”, then it is complete the moment you start. As soon as you put pen to paper you have “written some instructional documentation”.

On the other hand if the task is well defined and has a measurable deliverable then the goal is not achieved until it is delivered. For example: “complete a user guide and a technical manual”. This task definition is much more useful because it has a clear measure of success. The only time the task is complete is when the documents are written, have been reviewed and edited and are ready for publication. You are finished when there are no more changes to be made and you can move on.

The danger in believing that tasks can be partially completed is that it gives you a false sense of security. Because 50% of a task can be such a hard thing to define, people will tell you they have completed 50% of the task when they are 50% of the way through the time allocated to it. It might be the case that 90% of the task remains but they will insist that since there is 50% of the time left there must be only half of the work left, too.

This misconception is particular entertained by those people who believe that time is elastic. That is that you can cram any amount of work into a particular length of time, it just depends on how hard you work. These are the people that will tell you one of two things:

1. After working 10 days on a 20-day task they announce that they have only done 10% (i.e. nothing but 10% sounds better) and that they will make up the time. This is despite the fact that they fought tooth-and-nail in the first place to get 20 days allocated to the task. Truth be told they are simply rolling their delays from one task into the next and picking up speed as they accelerate through the project towards non-delivery.
2. The second case is that they report good progress all the way through the first 15 days of a 20 day task and then their progress slows down so that on successive days they go from 95% complete to 96%. They will certainly overrun their task and be claiming three weeks later that the last 1% of the task is nearly done. The old adage of the “first half of a task takes 90% of the time and the other half of the task takes the remaining 90% of the time” is all too often true.

It is much better to break tasks down into smaller steps and to simply report on their completion. Then if a particular task misses its completion date it automatically triggers an adjustment to the schedule. There is no ambiguity, there is no confusion.

# Managing People

It is not often recognised that a project manager must be a people manager as well. Often project managers come from a particular technical stream and they find themselves elevated to a leadership position without any formalised management training. Their first project may confront them with their first test in team leadership.

Most project managers therefore excel at the technical aspects of project management such as scheduling, design and testing. Many, however, are weak or uncomfortable with the core management disciplines which deal with 'soft skills'. This section will give an overview of some important people skills for the project manager.

## Negotiation

Negotiation can be a tricky business for technical people, we tend to see the world as a black-and-white, binary environment. 'Techies' often believe that there is a right and wrong way to solve a problem, or that one technology or solution is the 'best' available. This is part of their drive for perfection but in truth there are many ways to solve a problem and each technology or solution has its strength and weaknesses.

Negotiation is the process of achieving consensus while avoiding conflict. Central to this is the understanding that the best solution to a problem is one which attracts the consensus of all those involved. A unilateral (binary) solution is by definition not the best solution since it alienates or disappoints someone. Finding the best solution will involve compromises and the project manager will be the fulcrum around which the discussions between different parties revolve.

Most people view discussions as a zero-sum-game. That is, in order to "win" or succeed, someone must "lose". For example a salesman might believe that he will "win" if he can convince someone to buy a product at a high price. This is a zero-sum attitude, the salesman has "won" and the customer has "lost".

A customer on the other hand might not care about the price and might be willing to pay it if the product has the right features. If the salesman can work out what the customer wants he might be able to sell him the right product. Further if a particular product doesn't have those features, the salesman might be able to drop the price or offer other incentives that will convince the customer to buy. If he achieves this then they *both* win.

This is the art of negotiation.

By understanding that problems can be broken down into a number of elements which, when handled separately, produce trade-offs by which you can achieve a "win/win" solution. This is a solution where both parties walk away happy. This avoids the dichotomy of a binary, yes-no problem and the situation where both parties hold equally strong views, resulting in conflict.

This requires a leap of faith. If you approach discussions with the idea that negotiation is some tricky way to beat people and get your own way, you will certainly fail. People are not stupid and most will be able to discern your intent regardless of your outward demeanour.

Negotiation must be undertaken from a basis of trust and if people feel you can't be trusted then it doesn't matter how clever your approach. Try it out for yourself. Approach a discussion with an open attitude and work through to a consensus. Afterwards evaluate how you feel about the solution; you may be surprised. If not, get a job as a used car salesman.

## Understanding

Understanding is fundamental to negotiation. You must understand the proposal under discussion and the options available. You must understand what each party involved in the discussions seeks to gain from the discussion. If the discussion is composed of groups of individuals you should understand the goals of the individuals *and* the goals of the groups (which may differ).

You must also understand what you bring to the table and what you are prepared to concede. By knowing what you have to 'trade' you can enter the discussion with an open mind and flexibility. Ideally you should know this before you enter negotiations but sometimes this isn't possible.

## Empathy

Empathy is understanding the emotions of those involved. Emotion can cloud communication or it can enhance it but it cannot be excluded. As human beings we react to things on an instinctive emotional level and, like it or not, this dictates much of our reactions. By understanding the basic feelings of individuals in a negotiation it is possible for you to appeal to them on a more direct level than simple logic.

## Trust

Successful negotiations are built on trust. Without trust there can be no true compromise and therefore no solution. If the concessions you offer are insincere or grudging, then you will get an equally grudging response from other parties involved. Often it is the responsibility of the project manager, who holds a pivotal position, to take the lead by establishing a basis of trust between the parties involved. By setting the standards a project manager establishes the basis for negotiations and the tone in which they are conducted. This can often be done by a simple appraisal of the facts and an appeal for assistance to the parties involved. Something like "I understand we have a problem and I expect everyone here to contribute to the solution" often works wonders.

## Contribution

Once the fundamentals have been dealt with, a discussion of the problem usually ensues. The discussion should be conducted on a rational basis avoiding violent emotions. The problem should be clearly stated and agreed upon and solutions offered. During this stage of the discussion it is essential that *all* parties contribute. Silence is not acceptable. Comments should be sought from everyone, their concerns aired, addressed and hopefully resolved. Failure to do so will leave one or more parties feeling disenfranchised or disenchanting.

## Consensus

Once the various solutions have been discussed a possible solution can be proposed and if it is acceptable to all, taken forward. If not, some form of compromise must be hammered out and each party must be ready to conceding elements of their requirements in order to find a solution. Each party must then signify their willingness to accept the compromise and move forward.

Rolling over and sticking your feet in the air to have your tummy tickled does NOT constitute a win-win situation ! Don't enter into negotiations with the expectation that you will have to make all the concessions to reach a consensus. This will leave you feeling vulnerable and powerless.

This is known as a "win-lose" outcome.

You should enter into the negotiation with the expectation that everyone present will be able to give-and-take on their respective issues and work with them to achieve a suitable outcome. You should set your own minimum level of expectations and not be satisfied with the process until you have achieved it.

## **Building a team**

Wince if you must at the title of this section, but one of the most important facets of project management is “team building”. Rather than the fatuous team building games you often encounter at company days, I am referring to some more subtle people management skills.

### Trust – Be Open and Honest

Projects, like offices, can often be secretive places with various levels of disclosure due to commercial or political pressures. However the project grapevine works just as fast as the office kind and you can be assured that if you are keeping someone in the dark or deceiving them, you will be on the short end of office gossip faster than you can say “voluntary redundancy”.

Be as open and honest with your team mates as you can. Answer their questions directly and act as a conduit of information for them, not a barrier. If you feel you cannot divulge something, say so. Your team will appreciate your honesty and reciprocate by relaying information and producing honest and accurate estimates for you.

### Equality – Be fair and even handed

One of the maxims I have lived with as a manager is “individuals can succeed but only the team can fail”. Essentially this means that in public you should dish out credit wherever it is due but never criticism. Being criticised in public, in front of your peers, is not a motivating force for anyone.

If there is a project issue that needs to be addressed you can normally broach it as a subject for the whole team to address. By sharing the burden for issues, most teams pull together to solve the problem. By landing it on the shoulders of one or more individuals you often split the team and cause conflict. Open discussion of the problem will encourage the team to take ownership for the problem and solve it themselves.

### Loyalty – Protect your team

You will have a split responsibility - on the one hand you have a duty to your client to see the project succeeds - on the other you have a responsibility to represent your team and to support each other. Usually these two aims should be neatly aligned – but not always!

In a situation where you have to choose between the two you need to take the difficult moral stance. Don't air your dirty laundry with the client. Discuss the situation with your team mates and come up with a solution, present this to the client instead.

### Learn to delegate

The joke in armed forces often runs that the only order an officer ever need issue is “Carry on Sergeant Major!” Officers are expected to lead and leave the actual getting of things done to those more suited to the task, the troops. If you are dividing up work make sure you delegate properly.

Proper delegation entails laying out the task so someone understand its, so that it has reasonable and achievable goals and so that you give them all the support they require to get the job done.

It also entails giving them enough room to get the task done on their own. If you leave the execution of tasks to them they will, in return, leave you alone to get on with your job. If you spend you time looking over their shoulders it will only annoy them and waste your valuable time.

# Implementation

Hang on a minute, you're not finished yet!

Okay, the project's ready, development has finished and it's been thoroughly tested. But you still have to roll it out to a client who's going to use it in anger. This is the heart-in-the-mouth time when all your hard work is really put to the test.

Prior to launch you have a (relatively short) period to ensure that everything is ready to go and the your project is in a fit state to be released. Leading up to the launch you'll want to do some final testing and plan the release of the product into the live environment where it is to be used. A strong methodical process is important at this point so that you avoid any last minute hitches.

## Acceptance Testing

Acceptance testing forms an important and separate phase from previous testing efforts. Unlike the functional or technical testing, acceptance testing focuses purely on the "acceptance" of the product. Do the clients want it ? This is the pay-off in a commercial project, where the cheques are signed and the money changes hands.

If you have a formalised project you should probably define some acceptance tests earlier in the project, probably in the design phase and signed them off with your client. You can look at your (SMART) requirements and establish a set of tests from them that will prove your product fulfils its goals. This could be as simple as a checklist or as complicated as you like.

## Release Control

When you release a product 'into the wild' you will need to know what you released, to whom and when. This may seem trivial but you'll want to keep track of things so you can make fixes and changes as the customer discovers problems (and they will!).

These are the things you typically need to track for each release :

- The version number or identifier of the release
- The date and time of the release
- The purpose of the release (maintenance, bug fix, testing etc)
- For each component within the release
  - the name and version number of the component
  - the date it was last modified

Here is an example :

Version	Date	Type	Components			
			Name	Version	Last Mod	Hash
3.0.5.28	27-05-2004	Internal	Kernel	2.3.0.25	25-05-2004	#12AF2363
			GUI	1.0.0.12	27-05-2004	#3BC32355
			I/O Driver	2.3.0.01	01-04-2004	#B321FFA
3.0.5.29	05-06-2004	Internal	Kernel	2.3.0.28	03-06-2004	#2C44C5A
			GUI	1.0.0.15	01-06-2004	#32464F4
			I/O Driver	2.3.0.01	01-04-2004	#B321FFA
3.0.5.30	etc...	...	...	...	...	...

## Positive Perception

Most technical people are poor at the public relations side of business. Communications and marketing is generally not our strong point. But no matter how much we might like to think differently, people base decisions upon perceptions. Your project needs to create a positive *perception* as much as it needs to create a positive result.

To this end you need to capitalise on the successful delivery of your project and put together a “go-live” launch that suitably dazzles your intended audience. This might not be a black tie event with flowing champagne and a marching band, but there should be some event that punctuates the launch of your project. This is also a good way to reward your project team and underscore your appreciation for all the hard work. At the very, very least you should launch your project with a general announcement like an email to all the stakeholders.

Finally remember that launches take time and organisation and remember to plan and schedule them appropriately (with lots of contingency!). Since a launch will necessarily be a fixed date with lots of publicity it can be highly embarrassing if you miss that date or are forced to delay it. Early on in the project you can assign a vague date to the launch and then once you have completed your final testing and you are feeling confident you can announce a specific launch date.

Don't forget to have part or all of your team present during the launch process. This has two important benefits. First and foremost it gives them the recognition they deserve for all their hard work. Sure, you did your part, but the last thing you want to do is sit in your ivory tower claiming all the credit. Secondly it can be fabulous mileage for your PR campaign. To be honest, people would much rather talk to the individuals who slogged through the mud and got the project there by the sweat of their own brows than talk to management!

## Maintenance and Upgrades

If your project has any permanency whatsoever it will require maintenance and possible upgrades. While you will have done your level best to provide for all the users' requirements, those requirements will evolve over time and your product must evolve with them.

An important part of the maintainability is to make your project, design documentation and supporting material available as an important reference for those that follow in your footsteps. If they are to successfully implement changes within your system to cope with the future needs of users they must understand your thinking.

### Scheduling Upgrades

Upgrades are always necessary. You can either ignore them or plan for them ahead of time. If you acknowledge an upgrade or maintenance release will be necessary then you can plan one, say three months from the initial launch of your project.

This will prevent a hiatus in your project team since members can move from working on the initial launch to designing and specifying the next release of the product. They can start to ‘triage’ the defects that have been left from the previous release and decide which will be fixed in the maintenance release and which will not be.

Extending the idea further, if you are working in an environment where you will have continuous releases (rather than a one-off product release) you should contemplate scheduling regular point releases of the product. For example if you are developing a product that will have a major release every year you should consider scheduling a minor release every six months, or possibly every three months, for maintenance. A regular release schedule helps your team, customers and support personnel anticipate releases and plan for them

## **Support**

Your project has rolled out to a fantastic launch and you now have hundreds of eager users, but you're still not finished yet. What happens when something goes 'wrong' ? Who gets to explain to users how to use the product and configure it properly ? What if (heaven forbid) a user finds a bug in your product?

### Training

Simply dumping your product or system on an unprepared support team will not be appreciated. The people who are to support your finalised project need to be trained and prepared to do their job properly. As usual the scale of this training will vary with the scale of your project.

At the bottom end of the scale, provision of user manuals and support manuals will be adequate. If your product or system is not very complicated then that may be sufficient. More complicated products might require specific training guides and support material, such as tutorials or simulations to get the support team up to speed. At the top end of the training rung you might want to consider full blown training presentations and accreditation for support analysts.

Also consider using members of your project team to train support analysts. No one will know the product as well as the people who worked on it and some of these people will probably operate as third level support for the system once it has rolled out. By opening up these sources of information to the support teams you will gain their appreciation and support and your project team may actually enjoy getting to interact with the people that will be directly using the product.

Project management can be a difficult and thankless task. When things go wrong you shoulder the blame and take responsibility. When things go right the credit goes to the team and rarely do you get singled out for a pat on the back.

Yet projects comes with astounding rewards. Astute business managers know that it is not the cleverest technical specialist or the smoothest talking salesman who adds real value to a business. It's the people that can get things done. People who consistently delivers soon gains a reputation for solving problems and will be rewarded accordingly.

Having said that, getting things done is not always easy. Politics, resource limitations, technical problems, personal differences and organisational obstructions can all hamper your attempts to succeed.

The ideas in this book are not a new proprietary model nor a mystical insight into project management. They are merely an observation on how things actually work. It is drawn from real-world involvement in projects and a genuine frustration with existing methodologies.

Using the ideas in this book you should be able to plan, control and execute any project. It is a model that should be easy for the whole team to understand and, with a bit of luck, will allow you to deliver your project on time, on target and under budget.

Best of luck,  
Nick Jenkins.

Acceptance Test	Final functional testing used to evaluate the state of a product and determine its readiness for the end-user. A 'gateway' or 'milestone' which must be passed.
Acceptance Criteria	The criteria by which a product or system is judged at Acceptance Test. Usually derived from commercial or other requirements.
Alpha	The first version of product where all of the intended functionality has been implemented but interface has not been completed and bugs have not been fixed.
Baseline	A snapshot at a particular point in time of part of a project plan. A "schedule baseline" is a snapshot of the schedule at that point in time. Can be compared over time.
Beta	The first version of a product where all of the functionality has been implemented and the interface is complete but the product still has problems or defects.
Big-Bang	The implementation of a new system "all at once", differs from incremental in that the transition from old to new is (effectively) instantaneous
Black Box Testing	Testing a product without knowledge of its internal working. Performance is then compared to expected results to verify the operation of the product.
Bottom Up	Building or designing a product from elementary building blocks, starting with the smaller elements and evolving into a larger structure. See "Top Down" for contrast.
Critical Path	The minimum set of tasks which must be completed to conclude a phase or a project.
Deliverable	A tangible, physical thing which must be "delivered" or completed at a milestone. The term is used to imply a tactile end-product amongst all the smoke and noise.
End-user	The poor sap that gets your product when you're finished with it! The people that will actually use your product once it has been developed and implemented.
Feature/Scope creep	The relentless tendency of a project to self-inflate and take on more features or functionality than was originally intended. Also known as 'scope creep'.
Incremental development	The development of a product in a piece-by-piece fashion, allowing a gradual implementation of functionality without having the whole thing finished..
Milestone	A significant point in a project schedule which denotes the delivery of a significant portion of the project. Normally associated with a particular "deliverable".
Prototype	A simple model of a product which is used to resolve a design decision in the project.
Quality Assurance (QA)	The process of preventing defects from entering a product through best practice. Not be confused with testing which is done to remove defects already present.
Requirement	A statement of need from a project stakeholder which identifies a required attribute of the system or product to be delivered
ROI	"Return On Investment" – a ratio which compares the monetary outlay for a project to the monetary benefit. Used to show success of a project.
Show Stopper	A defect that is so serious it literally stops everything. Normally given priority attention until it is resolved. Also known as "critical" issues.
Stakeholder	A representative from the client business or end-user base who has a vested interest in the success of the project and its design
Testing	The process of critically evaluating a product to find flaws and to determine its current state of readiness for release
Top Down	Building or designing a product by constructing a high level structure and then filling in gaps in that structure. See "Bottom Up" for contrast
Usability	The intrinsic quality of a product which makes it simple to use and easy to understand and operate. Often described as the ability of a product to <i>not annoy</i> the user.
White/Glass Box Testing	Testing a product with an understanding of how it works. See Black Box Testing.