Project Management Control

Preface

Organisations whose primary business is doing projects employ professional project managers, have well worn project management procedures and are culturally project-friendly.

Organisations whose primary business is not project-based still need to do projects, but the management culture is attuned to managing everyday operations. People asked to manage projects in these process-based organisations may be accountants, IT professionals, engineers, medical professionals, etc. They are not first and foremost project managers. Yet they are expected to manage projects - even major projects that will determine the organisation's future - in an environment that may not be project-friendly.

Some project management books and methodologies start from the presumption that the project team is available and all team members will work full time on the project. In process-based environments this overlooks some of the major challenges that project managers face: for many team members projects are not their "proper job"; they may be assigned to a project for only part of their time and being assigned to a project may be viewed as a career limiting move; senior managers may not always give projects and project managers the support they need.

This project management book will also be of interest to non-IT people who find themselves involved in "IT" projects. "IT" in quotes because there is no such thing as an IT project, only business projects - some of which happen to involve IT.

Index

Chapter 1 - Introduction and Principles

Simplicity and humour may not be the first things that spring to mind when you think of project management. However, we will try to keep it simple and even have the occasional go at humour.

This short first chapter sets out some principles and introduces themes that will be expanded upon later.

What is a project? Definitions abound but usually boil down to something like this: a project is a one-off, temporary activity with a clear start and a clear end (though some projects never seem to end); it has full or part-time resources clearly assigned to it; it has a temporary management hierarchy that takes precedence over the company hierarchy; and it sets out to deliver something: an IT system, a new product or whatever.

Given the following four characteristics of projects:

- finite time
- people assigned
- clear roles and responsibilities
- things to deliver

Have you ever had this feeling about a project?

- not enough time
- too few people
- people not sure what they should be doing
- too much to do

What causes this feeling of sometimes overwhelming pressure? In its widest sense, a lack of planning. People drift into projects without properly defining or planning them then one third of the way through they begin to realise what it's really all about and the panic starts. Sometimes the end date is thrust upon them arbitrarily, but sometimes project managers voluntarily commit themselves to dates, costs and deliverables without troubling to define and plan the project properly, which is lunacy.

You'll never remove the pressure altogether, in fact that's not even desirable: a bit of pressure is good for the soul and gets the adrenaline flowing. But getting the pressure to a sensible, containable level is the aim of many of the techniques we will cover. Such that when you make a commitment you at least have a fighting chance of achieving it.

Another popular definition of a project is: the way change is achieved. Left to their own devices many things gradually improve, but to bring about significant change, say to a business process, a project is needed. However, to this definition we would add the word predictable: projects are a way of bringing about predictable change. That is, at the beginning of a project we should be able to predict cost, end date, deliverables and even something about the quality.

What helps make a project predictable? Clear scope, clear roles, clear plans, clear control mechanisms, clear reporting structures... in other words good project management. But here is a question for you: if you take the total cost of a project expressed, say, in man hours, what percentage should be spent on planning, controlling, reporting, etc. - i.e. project management? Five, ten, fifteen, twenty percent?

The answer is: it depends. Imagine a simple project being done by a few experts who've done many similar projects before. They may need hardly any control - perhaps only 5% of the total cost will go on managing the project. But by contrast imagine a large, complex project spread across many locations, staffed largely by novices who only spend part of their time on the project. A huge panoply of control may be needed to keep everything on track - perhaps even 35% of the project's total cost might be spent simply managing it.

Suppose we conclude for a specific project it will be appropriate to spend 15% of the project budget on project management. But the sponsor says "cut out all that bureaucracy and reduce the project cost by 15%!". How would we respond?

Do we have a choice between a project costing 100 with project management and a project costing 85 without it? We do not. The choice is between a project costing 100 with project management and one costing an awful lot more than that without it. (And without the controls in place we'd probably never know how much more!)

In other words project management is an *investment*: it results in the project costing less than it would otherwise cost. If you don't believe that it could be because your project suffers unnecessary bureaucracy masquerading as project management. One of the many skills of the project manager is knowing almost instinctively how much control to bring to bear, so that small projects are not smothered by unnecessary red tape yet enough control is applied to large projects to keep them on the straight and narrow. Some project management methodologies seem to militate against this essential flexibility. But we will air our prejudices on the subject of methodologies later.

Suppose you are asked to manage a project which on closer inspection you realise consists of 5 sub-projects:

- business process re-engineering
- IT software development
- Hardware installation
- User training
- Implementation and roll out

And you discover that each of the 5 groups involved has a different way of running their projects and they each use different project management terminology. Would you have a problem in running the programme? Probably you would. Ideally, every project in a company (throughout for "company" read "public sector organisation" or whatever is appropriate for you) would use the same project management approach and terminology.

This could be achieved by buying a PM methodology and imposing it, as is, on all projects. An alternative might be to have rules and guidelines which are attuned to

the needs of your company. And notice: rules and guidelines, not standards. With a big book of standards it's not always clear what's mandatory and what's optional and people tend to apply all of it just in case.

What would be in the rules, what would be in the guidelines? The rules need be no more than 16 pages of A6 (that's very small, shirt pocket sized) which, as far as any experienced project manager is concerned, state the obvious. For example, one of the rules might be: before each project stage begins risks must be formally assessed and significant risks presented to sponsor. Now if you're thinking: "well of course we do that!" you've probably been there, seen it, done it and got the project management T-shirt. But maybe you're thinking: "that makes sense, but I'm not sure we ever tell the sponsor about the risks...". Either way, read on.

Whereas rules are mandatory and enforceable, guidelines are optional and, a bit likes a restaurant menu, offer a list of things from which you can choose: forms, procedures, checklists, etc.

Though, faced with your first large project would you know which things to select? Quite possibly not, which is why it might be a good idea for the project manager to state what controls he intends to apply to his project and then have someone independent - we'll call them Project Support - review what's intended. Project Support might conclude: "you've got far too much bureaucracy intended...". Or they might say: "you've got nothing like enough control planned...". Or with luck they'll say: "that looks about right". If one of the first two, Project Support would help you devise a more appropriate set of controls for your project - which is why we call them Project Support rather than Project Assurance or (ugh!) Project Audit.

Process vs Project

In many companies there are, at the extremes, people who are process-oriented and those who are project-oriented. For example, walk into any bank and watch what happens there. Someone comes in to pay a bill. The clerk performs the task which takes maybe 30 seconds. Each task the clerk will do during the day will be triggered by an external stimulus "please can I pay this bill?" and each task they do during the day will be independent of every other task they will do. People who grow up in these process-based or operational cultures develop a management style appropriate for this reactive kind of work, which boils down to Just Do It and do it now and do it quickly. You may have heard this approach referred to as JFDI.

However, people who grow up in construction companies have a different experience. When they joined as a young bricklayer they laid bricks in accordance to the project plan - the customer does not ring up and say: "now lay brick number 735"! Work is not triggered by external stimuli but by the project plan. For executives in construction the imperative isn't so much to answer the phone within three rings but to get the housing estate built in the scheduled 3 years. These guys know all about planning.

The trouble is, trying to do projects in inherently process-based environments can lead to a clash of cultures which can cause many problems, though the cause of these problems often goes unrecognised. Have you ever approached a senior

business manager and said: "I need two of your people to work full time on this project for the next month to define in detail the business requirements that the new system will satisfy when we deliver it in eight months time." And received the reply: "You must be joking! I've got a business to run here, I can't afford to have two of my people taken off their job for a month. Come back in six months, maybe we'll be able to tell you then what we'd like the new system to do."

Sometimes people who have never been involved in a project have no idea why it's important to define requirements so far in advance of delivery. They are not being awkward or bloody minded, they just don't understand. You've tripped over a project vs process cultural issue.

If you ask IT people who have experience of large projects what helps projects succeed they will tend to answer: "Planning, planning, planning and more planning." But a process-based manager's reaction to this can be: "No! I manage two hundred people on the business side of my company. We don't feel the need for all this planning nonsense, we just get on with the work and do it. In fact it seems to me that planning is just an irritating, initial delaying tactic used by the IT department to avoid ever doing any real work." Now there's a culture clash for you.

However, you will find that if you take the trouble to explain to senior process-oriented managers what projects are all about, they are intelligent people, they will readily understand and will conclude that obviously projects have to be planned, business resources properly assigned, requirements defined at the outset, etc. But all too often nobody explains and the culture clashes persist.

In a pure process-based company the company hierarchy is primarily designed to manage those at the bottom who actually do the work of running the business (Fig 1):

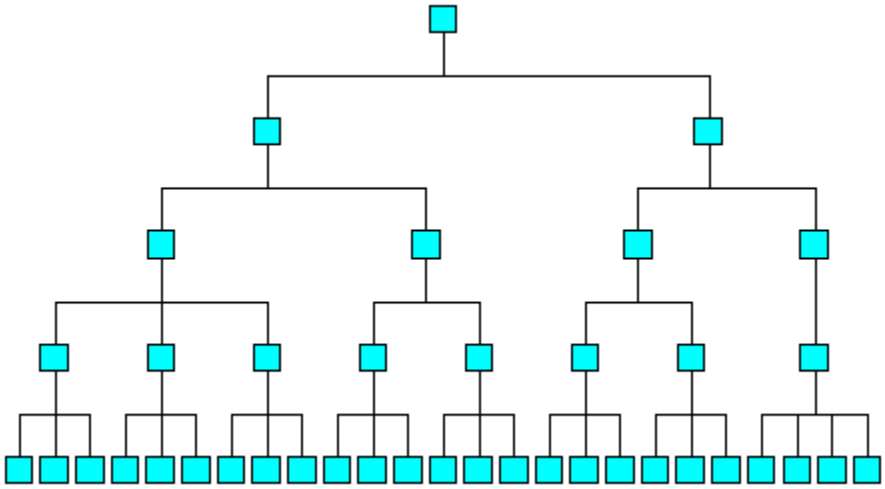


Figure 1: Process-Based Company Organisation

In the extreme you join the company, say, as an order-taker and there you stay along with 19 other order-takers taking orders for 30 years. Your line manager

manages his 20 order-takers and that's the limit of his horizon. We exaggerate to make the point.

However, in a pure project-based company things are different (see Fig 2). A resource pool of 50 engineers report to their line manager, 50 IT staff report to another line manager, 50 financial experts to another, and 25 project managers to another line manager. And they all sit around doing absolutely nothing, just sit twiddling their thumbs all day. Until, that is, a project comes along. Then a project manager is appointed. He selects 2 sub-project managers, 6 engineers, 8 IT staff and 3 accountants and fits them all into a one-off management hierarchy for the project. At the end of the project they all return to their respective line managers and sit and wait for the next project to come along.

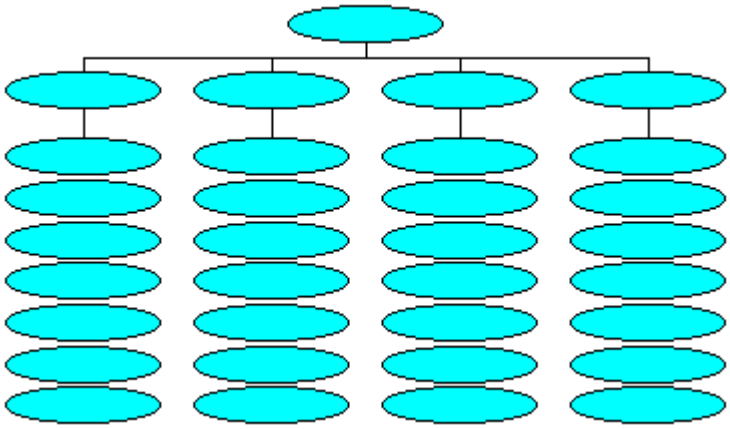


Figure 2: Project Based Company Organisation

In practice most companies are hybrids: some of a line manager's staff are doing work for him - their "proper job" - while others are rented out full time or part time to project managers. The line manager now has to be a task manager for those doing work for him, a career manager for those on projects and he also must manage the inherent conflict between servicing the day to day business and investing in the future.

In a hybrid company you could join, say, as an accountant: "Welcome," your boss says. "For your first three months you'll be working not for me but for this project manager." Three months later you return to your boss who then assigns you to another project for six months. In fact you could spend your whole career hopping from project to project and never actually do any work at all for your boss. (Though some would say this wasn't a totally unknown phenomenon even before projects...)

Which skill set tends to be in shortest supply in these hybrid companies? Is it accountants, engineers, programmers? No. Project managers. People who can manage projects in process-based cultures are worth their weight in gold. Many if not most projects in process-based companies are managed by people who do not see themselves first and foremost as project managers: "I'm an actuary managing this product development and development of the IT system that will support it." Yet these people are expected to perform well as project managers and overcome the

cultural project inhibitors that are often a feature of process-based companies - lack of empowerment of the project manager being just one example.

In some companies being assigned to a project is to be cast out into the wilderness - a vote of no-confidence from your manager, a career-limiting move and possibly a prelude to redundancy. And yet those left handling the day to day business see those assigned to a project as being "off on a jolly", escaping the pressures of the real world.

In other companies top management have established a more project-friendly ethos: "Projects determine our future. I want our best people assigned to projects. I want good project work rewarded with promotion. Indeed, henceforth, I will not appoint anyone to a senior management position unless they have managed at least one significant project." We will see how this state of grace might be achieved.

If you ask people what factors cause problems in projects these are the sort of things they will say:

- Undefined scope
- Unclear objectives
- Unclear roles and responsibilities
- Vague requirements
- Lack of leadership from sponsor
- Lack of user involvement
- Inadequate planning
- Scope creep
- Uncontrolled change
- Inadequate monitoring and reporting
- Team know it's impossible but management believe it will be done
- Ignoring reality, wishful thinking
- Inadequate communication

It is surprisingly rare for people to say that IT technology causes project failure or major difficulties. It is usually project management - or a lack of it - that causes the grief.

There was a touching belief a few years ago that if only one could write down every step a project must tread, then project managers could simply follow the process and out the other end would pop a successful project. Huge methodologies were spawned, money was made. But it isn't like that.

If you wanted to become a plumber you'd learn how to use each of the 50 tools in the plumber's toolkit. On your first, simple job you might only use 2 of them, but knowing which 2 and how to use them to best effect is obviously key. On a large plumbing job you might even use half the tools in your toolkit, but you'll never find any job on which you'll need to use every tool in the box.

The same with project managers: it's all about being equipped with tools for managing risks, defining scope, planning and scheduling, resolving conflict, etc and then using these tools appropriately if and when needed. You cannot manage projects by rote.

Chapter 2 - Projects and Stages

One day you're happily sitting at your desk when your boss walks up to you and utters some rather spine chilling words: "I have an opportunity for you."

It transpires that your company wants to write an all singing, all dancing new IT system that will replace all the existing systems and will run the company's whole business. Your opportunity is to manage the project. Having overcome your first reaction - to run like heck - you take a look and you realise that, at the extremes, the project could be done in two ways.

At one extreme you could establish a single, three year long project that would at the end of three years deliver the new system: the "big bang" approach.

But you realise it could be done another way: as a series of releases. Release 1 would, after 8 months, deliver a subset of the eventual functionality but would be installed and used in anger. Six months later Release 2 comes along and bolts on more function and by Release 4 everything will have been delivered.

Which might be the better approach? As always, it depends, but what might be the pros and cons of each.

These are the arguments usually advanced in favour of the 3 year long big bang approach:

- It's more visible
- More focus from senior management because it's big
- Only one implementation, one disruption to the business
- Only one tranche of training
- Plenty of time to do it properly
- More holistic approach to design
- Nice clean switch from old to new
- No temporary bridges from old to new
- Everything being done only once so lower cost
- If it's going wrong, plenty of time to get out before the end

And for the release approach these arguments are usually advanced:

- Better team motivation
- Less spent on project control
- Greater flexibility for senior management
- Learning from experience
- Easier to handle change
- Less business risk: evolution not revolution
- Less waste on never-used functionality
- Easier to manage four small projects than one large one
- Easier to commit people for eight months than for three years
- Fewer people leave during a release project than during the big bang project
- Earlier benefits realisation
- Lower cost overall (yes, it's on both lists)

Let's look at a few of these arguments in favour of releases in more detail.

Better team motivation. If you have ever worked on a long project you will know that at the beginning the team members tend to think: "We've got three years to do this, bags of time." And this lack of urgency can mean that not a lot gets done for several months. But if the end date is only eight months away there's much more natural urgency and the team think: "We haven't got long, we'd better get on with it."

Less spent on project control. The larger the project the greater the percentage of its budget will need to be spent on controlling it. So four smaller release projects may well in total spend less on project control than one large big bang project.

Greater flexibility for senior management. Imagine you are three quarters of the way through a big bang project. It's growing like Topsy and getting far too expensive. Can you this far through descope the project and save money? Probably not. Suppose you are having an eight bedroom mansion designed and built for you, it's three quarters built but getting far too expensive. Can you descope to a two bedroom bungalow and save money? No, that will cost you even more at this stage.

The same is true when you're building IT systems - beyond a certain point descoping will increase the cost not reduce it - not to mention the money wasted on what has been build that must now be thrown away. But with a release strategy, at the beginning of Release 4 senior management could simply decide not to do Release 4 and save its whole budget. Good news for the company but bad news if you happen to be the person whose need was going to satisfied by Release 4. There is always a plus and a minus, but at least with releases you get the choice.

Learning from experience. Release 1 goes live but the users conclude it isn't quite what they wanted. You can learn from that and adjust in subsequent releases. If the users had that same reaction when the big bang went live you have a bit of a problem. Equally, during Release 1 you will learn in project management terms and each subsequent release will be managed more efficiently and effectively than its predecessor, one hopes.

Easier to handle change. Suppose at the start of a three year long big bang project you define in great detail the business requirements that the project will satisfy. Obviously, over the three years a lot of those requirements will change - and change again, and change again. The longer the project the greater the percentage of its budget will be spent changing things that have been wholly or partially completed (see Fig 1). In fact some projects are so long they drop off the end of the chart: they never finish - so much has changed out there in the real world that the project is no longer relevant and it gets cancelled.

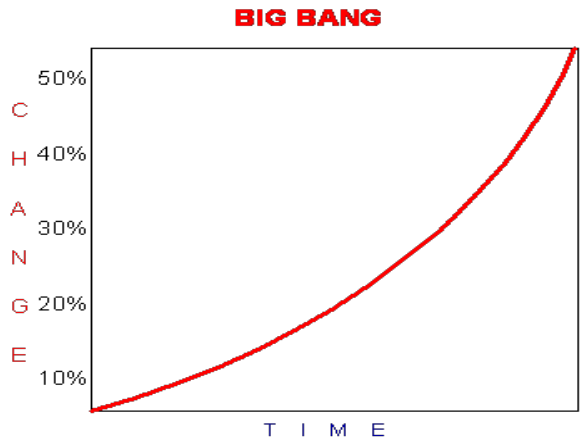


Figure 1: Percentage of effort spent on change in a long project

With releases you can much more readily react to changing requirements (see Fig 2). Yes, during Release 1 there will be some change to its requirements. But the real benefit is that at the start of Release 2 you define its requirements as the world then is, not as it was six months ago. And at the beginning of Release 3 you define its requirements as the world is then, not as it was a year ago. Release 3 may also be able to exploit technology that wasn't available a year ago - it's much more difficult to adopt new technology half way through a big bang. With releases you'll almost certainly spend less on change.

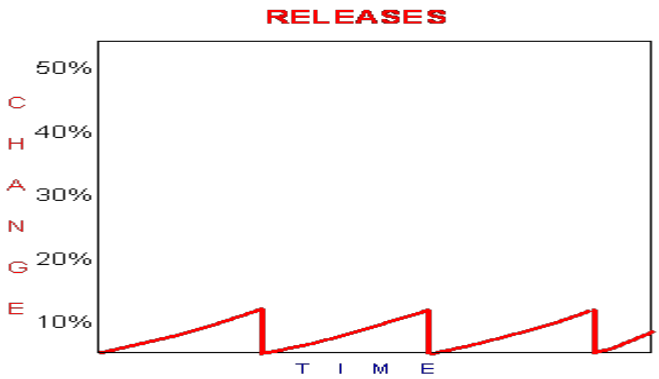


Figure 2: Overall less spent on change

As an aside, if you are at the start of a large IT project and your best guess is that 35% of your budget will be spent because requirements change as you go along, would you view the project as high, medium or low risk? Experienced project managers would probably say very high risk. Why? If you expect that much change you'll probably get more than that anyway and you may find that so much is changing all over the place throughout the project that it becomes very difficult to keep everything synchronised and you run the risk of chaos breaking out. Even if 35% of the budget is set aside to cover the cost of the change the change could still cause the project to get out of control. A good defence could well be to break the project into releases. If you only spend 10% - 15% or so of your budget because things change as you go along, you shouldn't have too much of a problem controlling change. We will cover change more fully in a later chapter.

Of course, with releases some of the cost of Release 3 could be re-engineering what had been delivered in Release 1, to improve it based on experience of use. This is either good news (you're improving things) or bad news (it costs you) depending upon your point of view.

Less business risk. The big bang project finally finishes. Cutting over the company's whole operation to the new system could represent a huge business risk - what if it doesn't work? Cutting over a bit at a time at the end of each release may help the Directors sleep a little easier.

Less waste on never-used functionality. Imagine you are a business manager and at the beginning of a three year long big bang project you are asked: "What would you like this system to do for you in three years time?" Your reaction will be: "Three years time! Well, we'll definitely need this and this, almost certainly this, probably these things, maybe this and we'd better put these things in just in case..." If you ever see a big bang delivered, albeit late and over budget, and you do a hard-nosed audit of which functions are being used and which are not being used you may be horrified to discover just how much of what you have delivered is never used. That is a huge waste of money. The world never turns out the way you thought it would three years ago.

Easier to manage four small projects than one large one. Small projects are usually easier to manage than large ones and will probably need a smaller percentage of their cost spent on project control.

Easier to commit people for eight months than for three years. It's much easier to lock someone in to a project for 8 months than for 3 years. And committing business people for 8 months probably means their skills will still be current when the project ends. After three years out of the front line they may need retraining.

Fewer people leave. The longer the project the greater the risk that people will leave during it. More people will leave during a three year long big bang than during a six or eight month long release. Anyone leaving a project team constitutes risk.

Earlier benefits realisation. The big bang project may deliver a lot of benefits - but only when it's finished. Three years can be a long time to wait. With the release approach you start to reap benefits as soon as Release 1 goes live. And who knows, these benefits might even pay for subsequent release.

Lower cost overall. Which approach will be cheaper overall, big bang or releases? Put your money on release. Less spent on control, less spent on change, less wasted on never-used functionality. And the cash flow advantage of earlier benefits realisation can have a powerful effect on the business case. Add to this the problems that large projects so often experience and the option of breaking it up into shorter sharper releases becomes yet more attractive.

In practice it depends upon so many factors that you can't say one option or the other will always be the better. If it were that simple people wouldn't be paid all that money to manage projects.

If you adopt a release approach, only include in Release 1 the functionality that will be needed on day 1 and leave all the other stuff and the nice-to-have features until later releases. But this raises one rather major question: whose job is it to ensure that Release 1 only contains the things that need to be in Release 1? People's answer to this question is often "the users" - which is a pretty meaningless answer, or "the sponsor" - which is true only to the extent that he is ultimately responsible for everything.

The correct answer is the project manager. This doesn't mean the project manager makes a random selection in splendid isolation. It means that the project manager makes the stakeholders (sponsor, steering committee members, users, IT, etc) work out what is really needed and what isn't really needed in Release 1. However, in managing that process the project manager may well find that he ends up knowing even better than the stakeholders what they really need and in the end he may have to, with the sponsor's blessing, make the scope decisions. It is ever so easy at the beginning of a large project to be everyone's friend and say: "You'd like that included? No problem I'll put that in for you." Anyone can do that. To be a good project manager you have to learn how to say the most important word in the project manager's vocabulary: "No". Politely but firmly: "No".

At the start of Release 1 cut its scope back to the bone - for which everyone will hate you. But they'll love you at the end when you deliver on time on budget. (Tip: having cut scope to the bone at the beginning, during the project slip in a couple of sexy features that you know everyone will like, but which you know will be cheap and easy to do - that'll make 'em smile).

There aren't all that many very large projects around so the chances are that the sponsor and project manager will not have been involved in a very large project before. This inexperience may mean that they drift into the big bang approach without too much thought (or even awareness) of the releases alternative and on the unconscious assumption that it must surely be cheaper to do everything just once. There are plenty of horror stories out there of large projects that started out big bang, went nowhere, then restarted with a release approach.

If nothing else, be alert to the danger that a big bang approach can imply, and the day the boss knocks on your door and says "I have an opportunity for you", from the very first moment assume the release approach. Then, if after careful consideration you conclude big bang really is better, fair enough. At least you'll be going in with your eyes open.

In summary, the release approach will probably cost less, deliver benefits earlier and be more likely to succeed. The release approach will give the business a better return on investment.

Let us turn to the other end of the spectrum - making small enhancements to existing IT systems. At the extremes there are two ways of dealing with them. At one extreme is the ad hoc approach: each time someone raises a request to enhance, it goes to IT, a programmer dives into the code, makes the change and it goes live whenever it's ready. The obvious benefit is that it is quick - assuming you have enough IT guys waiting around. Also, it doesn't require much management.

The alternative is to bundle, say, 20 enhancements together into a project - again let's call it a release. During a year perhaps there will be four such releases on February 1st, May 1st, August 1st and November 1st. The rule is that the live production system doesn't get changed between releases except where there is an overriding need. What might be the benefits of this approach? It's certainly predictable: the users know when each release is coming and can plan around them in terms of testing resources, procedural changes and so forth.

IT will probably do more thorough testing. There's an old adage amongst programmers: "it's only a one line change, it doesn't need testing." Into production it goes and crash, the live system comes tumbling down. You can't spend a week doing safety check, regression testing of a one line change. But if it's part of a quarterly release it's much more likely that there will be proper regression testing.

Economies of scale: it can take a programmer days to understand a complicated program before he changes it. If a number of changes are made at the same time the understanding overhead is borne only once. Doing those changes ad hoc would have meant re-understanding the program each time.

The sheer act of promoting updates to the live environment costs money - do it every quarter not every other day!

"If it works don't fix it." Every time a change is made to the live system there's a chance of a mistake. If the system is working acceptably leave it alone.

One of the biggest benefits of the release approach to enhancements is that some enhancements simply don't get done. Doesn't sound like a benefit? Consider this: how often does a user shout loudly "I want this changed, it's really urgent and important!" The change is made but a week later the user can't even remember why he wanted it. With a release approach there is more scrutiny of each potential enhancement and ones that aren't worth doing are weeded out.

Bottom line, it's cheaper to do enhancements as releases: you're trading the speed of ad hoc against the efficiency of releases.

Again, what is right for you will depend upon your environment. In a rapidly moving web-based retail environment updating the web site daily may be worth the cost, though even here grouping changes into a weekly update may be worth considering. If you look after the system that administers your company's pension scheme perhaps an annual release is best with changes between releases only if they are absolutely essential.

How does this release approach to enhancements work, what is the process? Sometimes it works like this:

- User sends enhancement request to IT
- User never hears anything about it ever again

We should be able to do better than that. Most IT groups have a service level agreement (SLA) with their users. One item in the SLA might be: within 3 working days of receiving a request to enhance we'll get back to you with a ballpark estimate.

This ballpark estimate enables the user to determine whether the enhancement would be worth doing given the cost. If so it becomes a candidate for a future release. But who decides which release, if any, it gets in to? All too often in the past IT decided and the perception was that IT didn't give the business what the business asked for and needed.

There is another way.

The IT systems that run the business are company assets. Assets need owners. Suppose the Marketing Director becomes System Owner for the company's order processing system. What responsibilities would he have?

- Account to the Board each year for the running cost of the system - this tends to stimulate efforts to reduce running costs
- Be accountable for the system's legal compliance - if the system puts the company in breach of the law somebody has to stand up in court.
- Be accountable for every penny spent enhancing the asset - decide which enhancements get done and which don't get done.

In a small company the System Owner can personally decide which enhancements get done but in a large company he will be too senior to do that so will delegate to his "Owner's Representative" - a less senior business person. The Owner's Representative prioritises the enhancement requests and, for example, recommends if only one could be done which one it would be. It becomes the Owner's Representative's job to recommend which enhancements get done in which release. Of course, genuinely urgent enhancements and fixes will short circuit the process and get done now as an exception to the "no changes between releases" rule.

Where the process works best the Owner's Representative meets informally with his IT counterpart every month or so to discuss what's coming down the track and what the priorities are. IT will suggest which enhancements it makes sense to do together for technical and cost reasons. Their discussion will include consideration of the IT resource plan (see Fig 3).

	THIS YEAR												NEXT YEAR											
	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D
BILLING																								
Release 10	2	2	1	1	1																			
Release 11	8	8	8	8	8	8	7	6	6	5	4	4												
Release 12							3	4	4	5	5	5	7	7	5	4	2	2						
Release 13													2	2	2	2	3	3	6	6	6	6	8	11
INVOICING																								
Release 2	2	2	2	2	2	2																		
Release 3						1	2	2	3	3	2	2												
Release 4													2	3	4	4	3	2						
Release 5																			3	3	3	5	5	7
PURCHASING																								

Release 1	10	10	10	12	12	12	14	14	12	10	10	8	6	4	1								
Release 2		1	1	1	1	1	1	1	2	4	6	8	10	10	10	8	8	8					
Release 3															3	4	4	5	8	8	8	5	4	4
TOTALS	22	23	22	24	24	24	27	27	27	27	27	27	27	26	25	22	20	20	17	17	17	16	17	22

Figure 3: IT Resource Plan

Imagine you are the Owner's Representative for the Invoicing system. It's March and you're contemplating the contents of Release 3 which starts in June and delivers in December. Currently 2-3 IT people are planned to work on Release 3. But you, the Owner's Representative, add up the estimates for everything you'd like in Release 3 and realise it would need 12 IT people to deliver everything you'd like as opposed to the 2-3 that are currently assigned. Who has the problem? You the Owner's Representative or IT? You do. It's your job to balance the contents of the release and the IT resources. It doesn't balance so what do you do? You'd consult with those who raised the enhancement requests and say "we can't afford to do all of these, which ones can we live without?" Note *we* can't afford to do all of them. Not "IT won't do them all for us."

By mid April you've pruned it, say, to 8 person's worth or work, but you're still 5 short. What might you try now?

You might approach the Owner's Representatives of Billing and Purchasing to see if they'll let you have some of the IT staff provisionally assigned to their projects. And what will they say? "We need more IT resources too..." Worth a try.

At some point you believe you have a strong case for more IT bodies to be assigned to Release 3 so you approach your System Owner.

The System Owners meet once a month to decide, inter alia, whether they want to re-slice the IT cake. Maybe your Owner wins you the 5 bodies you need. But if he doesn't, you have to cut the release content back to what can be delivered by those 3 IT guys. This is a business led process by which the business decides how they want to use their limited IT budget.

The business now have a nice plan showing the content of each release and the IT resources planned for each release. But what else should the business have a plan for? Can the IT people do these projects on their own? They most certainly cannot. Business users will need to define detailed requirements, be involved in and agree designs, do some testing, be trained in the new system features, and so on. Indeed, the user cost could be as great as or even greater than the IT costs.

But have you ever seen a plan which shows the business how much of their resource will be required for all these "IT" projects? Such a user resource plan often exists for a single large project, but a plan that adds up all the user resources for all projects and thus tells business managers how much of their resource will be needed for project work month by month is a rare thing indeed. And if nobody tells the Marketing Director that, say, 20 of his 100 people will be needed for project work what will he tend to assume in his planning? Zero. So every time a project manager needs a body from Marketing it's going to be an uphill battle.

There is a reason why project resources are not factored in to business managers' headcounts. Go back twenty or thirty years: there were many more people employed to administer the business and fewer projects, so the percentage of a business manager's resources assigned to project work was relatively small and didn't need to be planned specifically. Today automation means far fewer people administering the business and the pace of change means there are many more projects so the percentage of a business manager's resources assigned to project work is much greater and must be factored into business managers' headcounts.

It may even be appropriate to give one of the Directors an additional responsibility: Director of Change and Project Resource Planning. At its simplest this means asking project managers how much resource will be needed for their projects month by month (and if they don't know make them guess - anything is better than a zero estimate by default) and then add up all these resource requirements to show how many people (or FTEs - Full Time Equivalents) each business area will need to set aside for project work.

The first time the totals of business resources for projects (see Fig 4) is presented to the Board their reaction may well be "we can't free up that many from running the day to day business!" which can then lead to a constructive discussion about the organisation's capacity to do projects.

	THIS YEAR												NEXT YEAR											
	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D
BILLING PROJs																								
HR						1	1	1																
Marketing						1	1	1	1	1														
Admin	2	2	2	2	2	3	3	3	3	3	2	2	1	1	1	4	4	4	4	4	4	4	3	3
Accounts	2	2	2	2	2	2	3	2	2	2	2	2	2	2	2	2	3	3	6	6	6	6	8	8
INVOICING PROJs																								
Marketing											1	1	2	2	3	3	3	3	2	2	2	2	1	
Admin	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	2	1	1	1	1	1	1	1	1
Accounts	4	4	4	6	6	6	6	6	6	6	6	5	5	3	2	2								
PURCHASING PROJs																								
Marketing							1	1	4	4	4	4	4	1										
Admin	10	10	10	12	12	12	14	14	12	10	10	8	6	4	1									
TOTALS																								
Marketing						**1**	**2**	**5**	**5**	**5**	**5**	**5**	**6**	**3**	**3**	**3**	**3**	**3**	**2**	**2**	**2**	**2**	**1**	
Admin	**13**	**13**	**13**	**15**	**16**	**17**	**19**	**19**	**17**	**14**	**15**	**13**	**10**	**8**	**5**	**3**	**5**	**5**	**5**	**5**	**5**	**5**	**4**	**4**
HR						**1**	**1**	**1**																
Accounts	**6**	**6**	**6**	**8**	**8**	**8**	**9**	**8**	**8**	**8**	**8**	**7**	**7**	**5**	**4**	**4**	**2**	**2**	**6**	**6**	**6**	**6**	**8**	**8**

Figure 4: Business Resources for IT projects

If the Marketing Director has 100 people in his empire but the analysis shows that 20 will at any time be assigned to projects (many more than in the chart above where it's only between 1 and 6), he knows he really only has 80 people to do whatever Marketing does. He should have to account to his boss only for how he has used 80 people. Who accounts for how the other 20 are used? Those sponsoring the projects that use those 20. And indeed the Marketing Director's boss should make it clear to the Marketing Director that those 20 FTEs *must* be working on projects.

One could even consider having everyone in the company do time recording each week. If someone had spent all last week doing their "proper job" there could be a default time record, but if that person spent 10 hours user testing project X they must record those 10 hours against the project X time code. Then perhaps for the first time the company will begin to understand the true cost of projects as opposed to the IT tip of the project cost iceberg.

How does all this help us as project managers? Fairly obviously if you ask Marketing for a resource you're more likely to get a positive response if it's in the Marketing resource plan. It doesn't guarantee you'll get the resource or the right resource but it makes it more likely.

This is all part of the shift in company culture from process-based to a project-based or at least from a project-inhibiting culture to a project-enabling culture.

Let us now turn our attention to the software development lifecycle.

IT people love inventing new words and co-opting existing ones into impenetrable jargon in order, some would say, to bedazzle the uninitiated.

Extreme Programming, RAD (Rapid Application Development), DSDM (Dynamic Systems Development Method), Functional Specification, EFBERD, Object Oriented - the list goes on and on. Every few months a new way of doing software development is invented. Except that it isn't.

There are something like 400 activities in the software development lifecycle. Define a data item; write a program; write a test case; design an icon; hold a team meeting; etc. Whatever you call the process into which these activities are slotted, the software development lifecycle remains fundamentally the same. A rose by any other name... But renaming the rose gives the illusion of dynamism, of innovation, of the availability of easy solutions to thorny problems.

If only it were so.

Developing large software applications is tremendously difficult. Witness the number of well publicised software projects that are seen as very expensive disasters. If anyone really had found an easy answer to the software development challenge such disaster projects would surely not still be happening.

Unfortunately, problem projects are not caused by "a problem" which has "a solution". We will examine some of the factors which, if not addressed, lead to software project failure.

We have in a sense already alluded to one of these: the notion that there is such a thing as "an IT project". This conjures up the idea that a commissioning company or public sector organisation can call in a software house and get them to do the project: leave it all to them and let them get on with it. The suggestion that the commissioning organisation should commit significant numbers of their own staff to the project provokes the response "what are we employing the contractors for?" You don't buy a dog and bark yourself, do you?

Underlying this attitude is a fundamental non-understanding of the software development process. Talk to users who are involved in a software development project and you will find they often have no conception of the lifecycle - the manufacturing process - of which they are a key part.

So, let us look at the software development process and see how user and IT, client and supplier should be working together.

Whatever you call the following steps - and everyone and every methodology has their own terminology - you must go through these steps when developing software.

Project Definition Defining project scope, roles, business case, etc.
Requirements Analysis Defining in business terms the business data and business processes that are to be automated.
User Functions Design Producing a system design that tells the users what the system will look like and what it will do.
IT Technical Design Technical design of the system's internals, i.e. technical mumbo-jumbo that the user does not need to know about and wouldn't understand.
Build and Test Build could be writing a million lines of new code or could be modifying a package or an existing bespoke system, and then testing that it does what the User Functions Design says it should do.
User Acceptance The users say it's great and can go live.

Try this the next time you have a project team meeting. Give the team ten minutes individually to write down what work should be done, and to what level of detail, in each of the six steps above (or your equivalent of these six steps). Then get them to compare notes. (And by the way, "the team" includes the users.)

Will they all have the same opinion of the major activities and outputs of each step? You can bet your house they won't. The IT guys will have strong, but differing, opinions about what they think should happen in each step and many of the users won't even understand the question let alone have a clear view of the answer.

Imagine on Monday morning you're starting a project. At the peak there will be 60 people in the team. 30 IT and 30 business users. They've never worked together before and you ask them the question: what should happen in each lifecycle step. You'll probably get 61 different answers - 60 plus yours. Might this cause a problem in your project? It could cause chaos. What would you do about it?

One solution might be to get all the team members you can identify (at the beginning you won't know who all 60 will be) together for a day and define exactly what your software manufacturing process is going to be. At the end of that day you will have for each step a list of the activities within it and the deliverables from it.

For example, you might conclude that the User Functions Design step will deliver a User Functions Design document (Functional Spec, call it what you will) which will have five chapters and chapter 3 will be screen and web page designs (and you may even have a model from a previous project); a revised Cost Benefit Case; a Test Plan; a Plan/Schedule for the IT Technical Design step; etc.

But, you say, aren't these things all defined in the standards? Surely we don't need to re-invent the wheel for each project? And shouldn't the IT guys know what they're doing anyway? Maybe; yes you do; and probably, respectively.

Nobody reads standards and even if they did they probably wouldn't understand them - and that's assuming the standards actually define the software development process adequately. Every project is different - you at least need to tailor standard processes to meet your project's unique characteristics. Yes, the IT guys may know what they're doing within their own specialist sphere but may lack a clear grasp of the overall development process. How much does your average techie know about gathering user requirements? And those users for whom this is their first project? They won't have a clue what should be done in each step of the software development lifecycle.

So from that one day meeting you emerge with a clear manufacturing process that your project will follow. But far more important, by the end of that day your team will have a common understanding of the manufacturing process you will use. They know what work will be done in each step and to what level of detail and they know precisely what will be delivered from each step.

Requirements Analysis

In that one day session there will be a lot of debate about the level of detail at which the requirements should be defined in the Requirements Analysis (or whatever you choose to call it) step. Some will say they should be defined "at a high level" (which means nothing), others will say "in quite a lot of detail" (which means nothing) and others will say "in as much detail as possible" (which can mean anything you want it to mean), and some won't really know what you mean by "define requirements" anyway.

In how much detail *should* the requirements be defined in the Requirements Analysis step? By the end of that step you must have defined every item of data (e.g. Invoice Number): how many digits or characters long it is and its valid values (e.g. must be all numeric). And for each business process (e.g. take an order): which data items go in to it, what happens to that data (look ups, calculations, is it to be stored). And for each output (e.g. an Invoice): the data that must appear on it and how that data is calculated or derived.

One way to get across to the team, particularly to the business users, how much detail is needed in the requirements is to say: "imagine the computer has not been invented (which usually raises a cheer) and you have to write a procedure which somebody off the street could follow step by step and thereby run your business - in how much detail would you have to write the procedure?" At which point they will all go "phew" when imagining the huge amount of detail that would be needed.

For example

1. When the phone rings pick it up
2. Ask the customer for their customer number
3. Look it up on the customer number list
4. If they don't know their customer number ask for their name and address...

And so on for many many pages. And even considering the above you can probably think of half a dozen what ifs? that would need to be defined (what if it isn't a customer who is ringing...?)

Holding brutally rigorous requirements definition workshops, which may last for weeks, can be an effective way of extracting from the heads of the business experts exactly what the process is that they want automating. But you will find that when you press an expert in, say, Credit Checking to articulate *exactly* what the process is he doesn't know. (And if he can't tell you, the programmers have got no chance.) So, you stop the session, tell him to go and find out, and continue when he has the answer.

And you must define all the what ifs. What if the customer changes the order? What if we entered the order incorrectly? Simply defining the basic processes that would be followed in a perfect world where everyone gets everything right first time will leave you with a system completely unable to function in the real world.

Defining requirements thoroughly and fully can be a very painful process - a bit like ice breaking: you ram at the ice, get stuck, back off, ram at it again until finally you break through.

Don't forget the less obvious users like Legal and Internal Audit: they will have requirements too particularly if you're building a new, green field site system. Trying to retrofit things like audit trails can be tricky.

Users are absolutely hopeless at defining their requirements - almost as bad as IT people. So what's the answer? People - business analysts you might call them - who are trained how to extract requirements from users. These people know how to run those hot-house requirements definition sessions and ensure the results are clearly

documented. They will be very hard-nosed and insist that no detail is glossed over, that no-one is allowed to get away with anything other than a complete and detailed definition of their part of the business process. This is very difficult as the business process being defined may not exist today - it may be a new business process or a re-engineering of an existing process. Running these requirements definition workshops really is a highly skilled job.

The requirements definition sessions should not just be attended by users and business analysts. Members of the IT design team should attend too. Why? Three reasons: they can point out that the way a business process is being formulated would make it difficult to design and program and perhaps that an equally good way of doing the thing would be much easier to solution. Secondly, these IT designers have a very powerful incentive to make sure the requirements are properly defined because if they aren't they will have great difficulty in the design step. But most important, the IT people get a feel for what the users really want - what's important to them, how things should work. They will pick up a lot of useful insights which would never have come through from a dry business process definition document.

Even try saying this. "Team - users and IT - in the Requirements Analysis step I want you to define all of the requirements such that there would be no need for the IT team to ask any user about anything later in the project."

In a project, whose job is it to make the users define their requirements properly? The project manager's.

User Functions Design

A joint team of users and IT having fully defined and documented the requirements, we now move on to the User Functions Design step. The primary output of this step is the User Functions Design (UFD) document which essentially tells the users, the client, exactly what the system will do and how it will look. It is not just pretty pictures of screens and web pages: everything the system will do (edits, calculations, etc) is specified.

Whereas in the requirements document everything was defined strictly in business terms, the UFD describes the IT system (or systems). Every edit, every calculation, every process mentioned in the requirements document is reproduced here in the context of the system design. If the requirements document says a field must be all numeric, that validation check would be described in the UFD as part of the logic behind the screen or web page on which that field is to be entered. Requirements that will be excluded from the design and eventual system must of course be listed with the reasons for their exclusion.

Vitally, all of the requirements that will be satisfied in the delivered system must be incorporated into the User Functions Design (UFD) document.

Much of the evolution of the design should be a collaborative effort between user and IT people. For example, the requirements document would specify the, say, 23 data items that comprise an order. The User Functions Design step will determine whether these are all to be entered on one screen or on several screens, or even on 23 screens. This has to be a conclusion reached both on technical grounds and on usability grounds.

Techniques such as inspection, prototyping and simulation can be adopted to ensure the UFD is complete and correct. We discuss these and other techniques for ensuring the quality of the UFD in the Quality Management chapter.

You might like to ponder this question (which will be answered in the Project Organisation chapter): who should sign off the User Functions Design document and be held accountable for its correctness?

There will be several other outputs from the User Functions Design stage: testing strategy, implementation plan, project plan for the IT Technical Design step, a revised cost benefit case, and you can probably think of many others.

Having finished the UFD document and got it signed off what happens when, later in the project, somebody wants to change a requirement? Which document(s) do you update? You could update both the requirements document and the User Functions Design document. But this clearly has a cost. We would recommend that only the User Functions Design document is updated. In other words, once the User Functions Design document has been completed, the requirements document is archived. Though frankly it doesn't really matter which document or documents you decide to keep updated. What *does* matter is that you and the team are crystal clear about which document(s) you *will* keep up to date. This is one of the many decisions you must make at the start of the project, perhaps in that one day meeting in which you hammer out your project's manufacturing process. As with so many things in projects, it doesn't matter how you do it as long as you and the team know which way you're going to do it.

IT Technical Design

The User Functions Design document tells us exactly what the system must do. The IT boffins can now get to work designing the system's internals: objects, databases - the technical mumbo-jumbo. In theory the users have no involvement in this step. In practice the User Functions Design document may not be quite as clear as we would have liked and questions of the users may need to be asked. Further, the techies might come to the conclusion that something specified in the User Functions Design document is just about impossible, or would at least be very expensive to deliver as designed, so they may seek the users' and designers' permission to change it. So the users do need to be on hand during what appears to be a totally technical step.

It's a good idea to have some of the people who will do the IT Technical Design working on the previous step, the User Functions Design: they will spot at that stage things that will be difficult to do technically.

In parallel with this technical design, the users should be engaging in another set of major activities. The users know what system they are going to get and they know when they are going to get it, so they can now begin preparing for user testing, user training, implementation, roll out, etc. These activities should all be an integral part of the project plan.

Build and Test

Churning out code in accordance with the IT Technical Design should be relatively straightforward as long as the requirements and User Functions Design are not being constantly changed and the IT Technical Design was properly done. By contrast, if the requirements and User Functions Design were woolly and/or keep changing, the Build stage can be chaotic.

How much business knowledge do you need to do system testing, assuming there is a high quality and stable User Functions Design document? Not a lot. If we have updated the User Functions Design document as changes came along during the project it provides an unequivocal benchmark against which the system can be tested. More about testing in the Quality Management chapter.

User Acceptance

Note, we do not call this step User Acceptance Testing. In theory no separate user testing should now be needed. We *know* the system works because System Test proved that it did. In practice it is seldom like that. The users do tons of their own testing and find a mountain of bugs. Is there any other industry in which the customer is prepared to spend a lot of time and money getting the bugs out of a shoddy product delivered to them by a supplier?

And one hopes the days are long gone when User Acceptance Testing is where the users start to redefine their requirements.

The users having accepted the system, we go live, pick up our large bonus for such a successful project and live happily ever after.

It is ever so easy in a software project to claim that the requirements have all been defined when they haven't really and move on having apparently met the milestone. It's equally easy to "finish" the design stage and claim the credit when the design is nothing like complete or correct. Then the proverbial spectacularly hits the fan in the build and test stage and/or when the system goes live. Please don't do it.

Lifecycle

Clearly, we have not sought to describe all 400 or more activities that comprise the software development lifecycle. We have given an overview of the lifecycle and laid stress on the need for user and IT people to work together as a single team, particularly during the Requirements Analysis and the User Functions Design steps. Having the users in isolation write requirements and then give the document to IT who then in isolation write a design document and pass it back to the users for sign off is a recipe for disaster. And getting the users to write the requirements *and* do the visible parts of the system design (such as screens, web pages and output formats) and pass that document to an IT organisation and say "deliver that" will probably be an even greater disaster.

It is unfortunate that in small, simple software development projects almost any approach will work. The users *can* define their requirements in isolation and pass them to IT and things will probably work out. The users can even do the design

themselves and pass it to IT. Iterative prototyping, DSDM, Extreme Programming - these approaches will all work in certain circumstances - particularly in small, simple, self contained projects with an experienced team. Unfortunate because people's experience that these approaches work means they try them on large, complex projects and find out the hard way that they may not scale up.

Incidentally, techniques such as RAD and DSDM are actually highly structured techniques that work well where applicable. It is when they are misinterpreted as being techniques that require no project definition, no planning, no documentation, no sign offs, etc that they become dangerous.

Whether you call your development process Waterfall, RAD, DSDM, Extreme Programming or whatever really doesn't matter. What matters is that whatever process you're going to use the team understand it.

Whose job is it to ensure that all members of the project team understand the manufacturing process that the project will use? You guessed it, the project manager.

The Analogy

The software development process is complicated and not easy to explain to senior managers. If you have only five minutes to get some key points about the software manufacturing process across to senior people try this.

Do you buy a ticket for the Lottery every week? Well, this week you're rash - you invest your pound and buy a ticket.

And at last your numbers come up! You haven't won a fortune though, only a million pounds/dollars/euros. Clearly you now have a problem: deciding what to do with the million.

You consider many high level solutions to the problem: buy a small island; retire; buy a yacht; invest it; etc. But in the end you decide the outline solution to the spend-a-million problem will be to have a house architect designed and built.

Your Project Definition Document (PDD) says: Problem - spend a million; Outline Solution - design and build a house; Sponsor - you; Users - your family.

You and the family spend many happy evenings defining your requirements. Now, secretly you have always wanted seventeen children - which you can now afford - so the requirement is for a house with 17 small bedrooms.

If you now approach a useless architect he will design a house for you which has 17 small bedrooms - and nothing else. Kitchen? You didn't ask for a kitchen. Bathroom...?

But a good architect will work collaboratively with you during the design step and take you back to your fundamental requirement which is to house the tribe, and working together you will design a house with a kitchen, bathroom, living room and perhaps 9 double bedrooms. Ideally you would perhaps have involved the architect

in your requirements analysis sessions so that he could have got a good feel for what sort of house you're really after and so he could guide you away from asking for things that would be impossible or ridiculously expensive to build.

Half way through the design step the architect says to you: "Look what arrived in my post this morning, it's a brochure advertising solid gold bath taps. Aren't they lovely?" You take one look and it's love at first sight: you must have solid gold taps throughout the house!

"But," the architect says. "They are £/$/€50,000 a pair!" In that case, you decide, you will change your requirements: you will only have two children and instead spend the money on solid gold taps throughout the house. Doubtless a wise decision.

This happens in all IT projects: when the business users see what is possible technically they say: "ahah, we didn't know that was possible, we'll change our business process to exploit that." You need to allow for this kind of thing in the design step plan.

A high-tech architect will now put the draft house design into a virtual reality simulator. You will don the headset and walk around your house in virtual reality, getting a feel for what it will be like when eventually built.

Even if you get this VR simulation the architect will still produce a document for you with all the floor plans, which way the doors open, whether the taps will be solid gold or gold plated, etc. And once you have signed off that document that will be the house you're going to get. If it was your house and your money would you pay attention to that document before signing it? You bet! Unfortunately users in IT projects don't have quite the same incentive to scrutinise the User Functions Design document. In an IT project whose job is it to make the users apply their minds to reviewing the UFD properly before signing it off? Once again you've guessed it - the project manager.

The techies now get to work on the technical design of your house. Would you be interested in knowing how many British Thermal Units your boiler was rated at or the Newton value of the concrete blocks? You wouldn't.

In some IT projects IT produce a design document that contains the user functions design and the technical mumbo-jumbo design muddled together and then ask the users to sign it off. Result: befuddled users. Don't do it. Keep the User Functions Design document to those things the user needs to know and put all the techie stuff in a separate IT Technical Design document.

Back to our house. Half way through the technical design step the builder calls you and says: "you know we promised you a twelve foot wide living room? Could we make it 4 metres instead, which is thirteen feet one and a half inches? No extra cost." You ask why. "Well, joists all come in metric sizes now, so it's actually cheaper to build it 4 meters than 12 feet."

You might say: "Great! More space no cost? Go for it." Or you might say: "No, the living room must be twelve feet wide because all my Persian rugs are 12 feet wide."

The same happens in IT projects. IT discover it would be much cheaper if two input screens were combined into one - would the users like to agree to such a change?

The builder builds the house and the builder checks that it all works in accordance with the "User Functions Design" document plus any agreed and signed off changes. How would you feel if, house nearly finished, the builder in the middle of winter invited you to come and live in the house on a trial basis to test whether the central heating works or not? You wouldn't do it, would you? But for some strange reason users of IT projects seem happy enough to test half-baked IT systems.

But your builder is a professional and gets the house right. You then move in and live happily ever after.

Two things are obvious about the house-building project which are equally true in software projects but not equally obvious in software projects. Firstly, in the house-building project where would you and your family be primarily involved?

- In the Project Definition step: it's a house not a yacht or an island
- In the Requirements Analysis step: we want 17 bedrooms
- In the User Functions Design step: the house will be like this...

In the Technical Design step your involvement is minimal (though you do need to be available to consider changes proposed by the builder), ditto in the Build and Test step and the Acceptance step should be just a formality. (Would you say; "ah, now I see it built what I actually wanted was..." Don't think so.)

In the house-building project you will be heavily involved in the project definition, requirements and the design steps. Exactly the same should be true for users in IT projects and for the same reasons. Though you do have a fairly major sub-project that runs in parallel with Technical Design and Build: the House Move sub-project. Getting bills and bank statements redirected, organising the removal men, etc. Just like the user implementation activities in an IT project.

Secondly, at the design stage you'd like a window moving three feet to the left. How much will the change cost you? Not very much, it might even be free. But when the house is nearly finished what will that same change cost you? A lot of money! The same is true in software projects but nothing like so obvious.

The analogy makes a couple of things very clear to senior managers: why they should invest their user staff in the early stages of the project; why they should ensure the design is checked very thoroughly; and why they should not change their mind about the requirements having signed off the design. Suddenly it's all rather obvious.

Designing and building houses is in many respects very similar to designing and building software and the analogy can help to de-mystify the software development process.

Though this notion that we should explain to senior business managers what software development projects are all about does rather overlook the Medieval Monk tendency amongst some IT suppliers. Medieval monks didn't like peasants to be able

to read as this enabled them to acquire knowledge and thus potentially challenge some of the nonsense with which the monks held them in thrall. For similar reasons some who later translated the Bible into English were burned at the stake (e.g. by Henry VIII): can't have the peasantry challenging our interpretation of the scriptures, bang would go our power! Interestingly when Henry VIII later dissolved the monasteries he commissioned an English translation of the Bible exactly for that reason - to erode the monks' power.

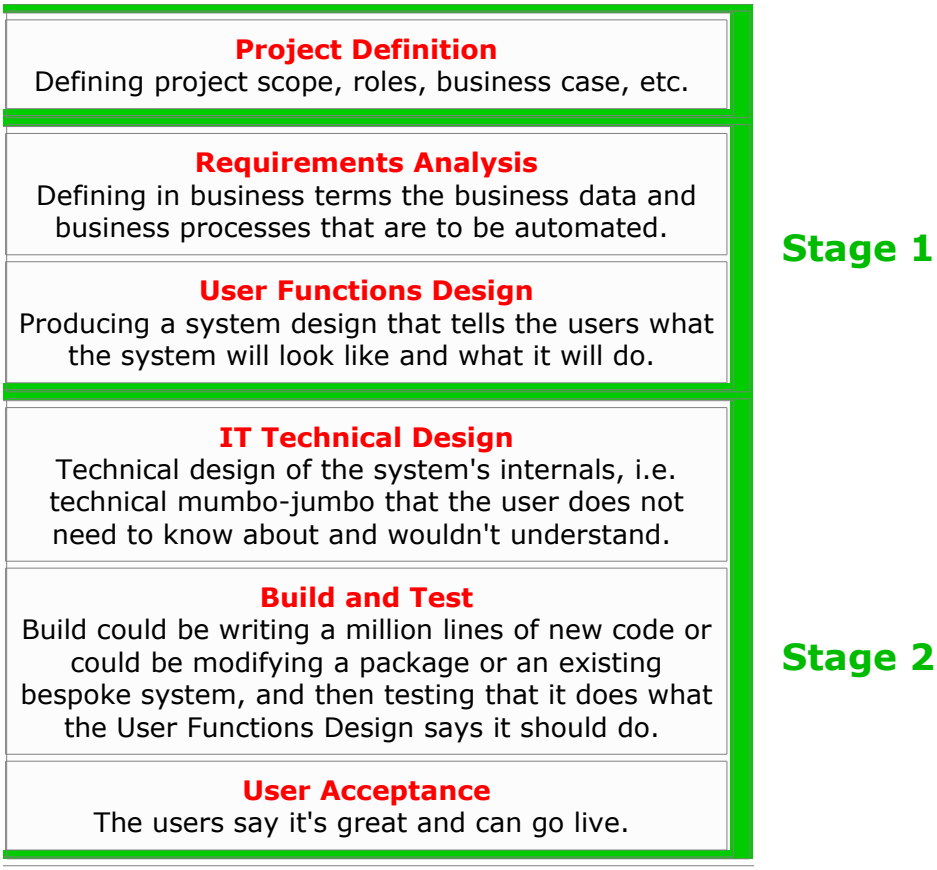
Projects are all different. Decide what's the most appropriate way to do yours, make sure the team understand it and enlighten senior managers.

Manageable Chunks

We come to the last bit of this chapter (fear not, most other chapters are shorter). Manageable chunks or more formally Management Stages. You need to grasp this next bit or you'll be all at sea for the rest of the book.

Suppose you have a project that's five weeks long and employs you and one other person. Having spent a couple of days defining and planning the project would you feel reasonably confident about committing to the cost and end date, give or take a bit? Please say "yes". How about if you were the sponsor, would you be happy to give the project manager the whole budget for that little project in one go and say "see you when you've done it?" You would. From a control point of view our five week project has one Management Stage.

But your next project is rather different. It takes ten weeks just to define and plan it and you realise it will require many releases and Release 1 alone will take about 12 months and will employ around 100 people at the peak. At the end of the project definition stage would you commit head-on-the-block to the cost and end date of Release 1? Probably not. You might conclude that you can only really estimate and plan with confidence up to the end of the User Functions Design step, and only when you have done the UFD could you plan and commit the rest of the project. The project looks like this:



The project consists of two Management Stages. (And arguably three: if the Project Definition step is going to take ten weeks we might well want to manage that as a formal stage too.) You can now see why we have hitherto been using the term step (Requirements Analysis step, UFD step, etc.): to differentiate them from Management Stages. In our project the Requirements Analysis step and the UFD step make up Management Stage 1. Equally stages are not the same as releases: in our example Release 1 consists of two Management Stages (three if Project Definition will be a formal stage). From hereon we will usually refer to a Management Stage simply as a stage.

Before each stage begins the project manager commits to the cost and end date of that stage and gives an outlook for any later stages. Before each stage begins risks will be formally assessed, a detailed plan and schedule for the stage will be built, roles will be defined and assigned and resource commitments for the stage will be obtained. At the end of the stage the project manager must re-evaluate project costs and benefits and go to the sponsor for authorisation to do the next stage.

How long in elapsed time should a stage be? If stages are very short you'll spend your whole life assessing risks, defining roles and getting authorisations and get completely bound up in red tape. But by contrast if stages are too long you'll be trying to predict the unpredictable. As always in project management the answer is: it depends. Many factors will influence how long stages should be. But perhaps something between 2 and 6 months would be reasonable with stages typically 3 or 4 months long?

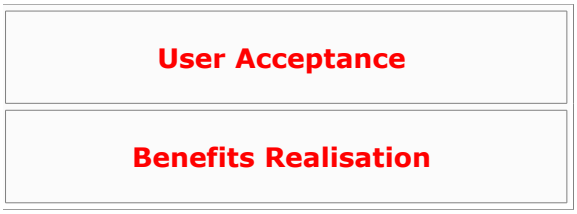
One might say that before each stage begins the project manager makes a fixed price contractual commitment to the project sponsor. We are beginning to apply the same sort of disciplines to internal company projects that would be applied to outsourced projects.

If at the start of Stage 2 the sponsor does not authorise Stage 2 in writing what is the project manager obliged to do? Stop work and send the team home. And wait for the sponsor to make his mind up either to commit the funds for Stage 2 or to cancel the project. This is not so much a discipline imposed on the project manager as on the project sponsor, to force him to make an informed business decision: is the project still a good investment for the company or not. Arguably too few projects get cancelled in this way, they roll on under their own momentum like a giant snowball even though the need for the project has long since melted away.

This principle of dividing projects into Management Stages for control purposes, and most of what follows in this book, applies to just about any sort of project that goes through steps such as design and build. If you wanted a new theatre built in your town you might get a fixed price quote from an architect to produce all the designs, then you'd get a fixed price quote for the construction.

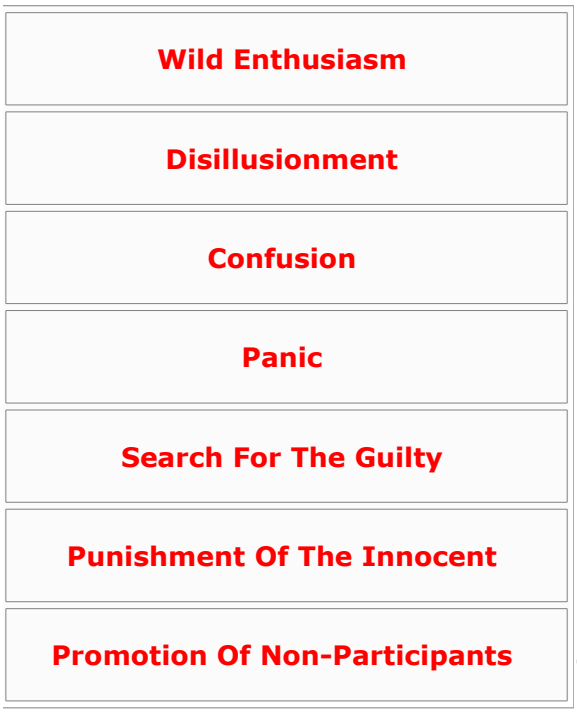
(If this was a methodology textbook we would now invent a fancy name such as "Linear Contiguous Stratification" for this simple idea of dividing a project into Management Stages. Not only does it sound impressive, it would also mean we could set an exam which asks "what is Linear Contiguous Stratification?" and if you picked the correct multiple choice answer we could accredit you in our methodology and you could claim to be a qualified project manager. We could even trumpet to the world that we had a "new" way of managing projects, whereas what we actually would have done is coin a new term for a universally used technique - which is essentially what all methodologies do. We will resist the temptation.)

In conclusion, if the project team understands the work steps the project will use - the manufacturing process - and they understand the control mechanisms wrapped around that work you have a fighting chance your project will look like this:



But if the team don't understand, your project will look more like this one:

Business Need ---->



----> Unhappy Users

Though we are quite sure you will not recognize any of these characteristics from your own experience. This kind of disaster project usually has one simple underlying cause: the project manager didn't do his job properly.

"A user is someone who tells you what they want the day you give them what they asked for."

Chapter 3 - Roles and Responsibilities

What makes projects such fun is that they are all different. And because they are all different each will need a unique project organisation.

If roles and responsibilities are not clear many problems will result:

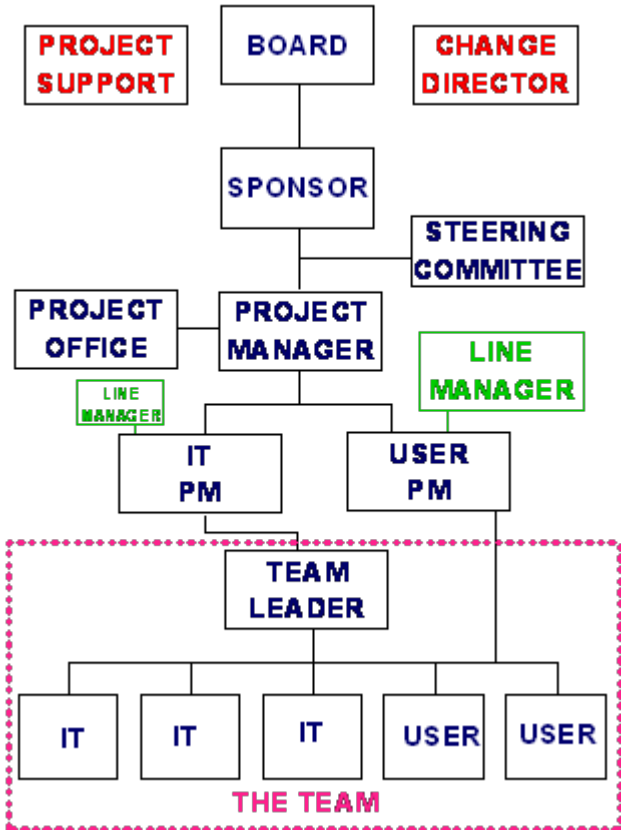
- ultimate source of authority unclear: poor leadership
- people do not know their responsibilities: people will not do what they need to do to make the project successful
- not clear who can decide what: slow decision making
- others' roles unclear: you don't know who to talk to, poor communications
- unauthorised projects start: nobody's job to authorise projects
- lack of accountability: there's little incentive to do things properly
- resources aren't committed: nobody is held to account for breaking resource commitments
- unclear objectives: project objectives are not owned by anyone, everyone has their own opinion, moving target, potential failure

So, before any project begins we must ensure amongst very many other things:

- a clear management hierarchy exists for the project
- each person has a defined and agreed set of responsibilities
- people will be held accountable by their line manager for performing their project role

Conjure up in your mind a medium to large IT project (or another type of project if you wish). If you feel energetic, find a piece of paper and draw a project organisation chart.

Is yours anything like this one?



Sample project organisation chart

Yours is probably quite different. Does that mean that yours is wrong? No, this isn't a methodology textbook. Indeed, there is no right answer: the organisation chart will be specific to your project.

Let us explore each of the roles in the example above and then consider some other possible project organisations.

One hopes your organisation chart at least had a Project Sponsor. Ever had the experience of asking a sponsor what his role is and getting a rather blank look by way of reply?

Role of Project Sponsor

The first thing to note is that on our chart the box at the top, Board, means Board of Directors. As far as the project is concerned the sponsor is top of the tree. But what is his role, what are his responsibilities?

- Accountable for the project. If it goes wrong the sponsor should be the first person up in front of the firing squad. In companies where the Board of Directors hold sponsors accountable, sponsors suddenly become much more interested in who is managing the project for them - their fate is partly in the hands of the project manager.

- Select, or at least approve, the project manager.
- Acquire funding for the project from the Board of Directors, or whoever it is that authorises funds for projects.
- Champion the project, promote and support it in high places.
- Own the project's business case.
- Accountable for realising benefits once the project is delivered. What effect might this have on prospective sponsors if they know they will be held accountable for the claimed benefits being realised post project? They will probably be a lot more realistic about the benefits they claim in the business case.
- Give the project manager the go or no go at the start of each project stage.
- State and own the project's objectives, have the vision (but beware, there's a fine line between vision and hallucination.)
- Make a presentation at the project kick off meeting, explain why the project is important.
- Act as the project's Godfather, that's Godfather in the Mafia sense: if there is a problem the project manager can't sort out, the sponsor has a quiet word with the offending party and makes them an offer they can't refuse. One hopes the problem will evaporate away, and that next time people will listen to the project manager. Indeed, it is not totally unknown for project managers artificially to engineer a crisis in the early days of a project, identify a guilty party who is on the periphery of the project, and then get the sponsor to sort that person out very firmly and make sure everyone knows about it. The message that goes out is then very clear: get in the way of the project manager and the sponsor will come down on you like a ton of bricks. This can encourage co-operation with the project manager. (Handle with care.)
- Resolve issues the project manager can't.
- Ensure only needed function is delivered and that money isn't wasted on unnecessary functionality and features.
- Accountable for legal compliance of the finished product.
- Chair project steering committee.
- Empower project manager to manage the project.
- Commission post implementation business case review.

How much work is involved in being a sponsor - how many days a month would it take? It should only be one or two days a month. If you think about it, there are some fairly heavy accountabilities on the list but not much day to day doing - that is delegated to the project manager. Of course it may be more than a day or so at the start or if major problems arise.

How many projects can one person sponsor at a time? Three or four, maybe five? But anything beyond that and the sponsor isn't going to be interested in, or even aware of, some of their projects and they become a name in a document and nothing more.

The trick is to have a sponsor who is senior enough to have the clout your project needs, but junior enough to be interested in your project if it's a small one. There could be several levels of director/manager in your company who can take on the role of project sponsor, the level of the sponsor for any given project depending upon factors such as the project's budget, criticality and the degree of cross-functional involvement. (If HR, Marketing, Engineering and IT are all heavily involved you'll probably need a senior person to be the sponsor.)

How would you feel about having several people acting as co-sponsors - good idea? No: you'd never know who was in charge. But imagine a project comes along and three members of the Board of Directors will benefit equally from it and they are each in effect paying one third of the project's budget. Rather than have 3 co-sponsors, one of the Directors becomes the sponsor. But if you were one of the other two Directors would you be happy to be shut out of the project altogether? Probably not: you'd want to see how it was going and to make sure your part of the business was getting a fair crack of the whip. We have just defined the project steering committee (sometimes known as the project board or even project review board).

Role of Project Steering Committee

Nominally the steering committee comprises the heads of those parts of the company (or companies) that are significantly involved in or affected by the project. So, if it's a large IT project for Marketing, perhaps the Marketing Director would be the sponsor and the IT Director would join him on the Steering Committee. In our example in the previous paragraph, the HR Director might have been the sponsor with the Finance Director and Logistics Director joining him on the steering committee, and let's assume the IT involvement isn't that great and therefore the IT Director will not be on this project's steering committee.

But who decides who should be on the steering committee? In some companies the Standards do! For example they might say there must be a sponsor, a senior customer and a senior supplier. So guess what? Every project has a 3 person steering committee (project board) whether it needs it or not. Who should decide who's on the steering committee? The sponsor should, but the project manager should make strong representations and suggest who would add value, who would actually help the project to succeed - that's why they're there, after all. Also agree with the sponsor when the steering committee should meet: the frequency may well vary during the project's life, and agree the mechanics of steering committee meetings: who will chair them (the sponsor), who presents what, etc.

A steering committee can be a menace: they squabble, cause delay by not making decisions, meddle in the detail - e.g. what colour the heading on a web page should be. (Sometimes called the bicycle shed syndrome: the company board have no strategic ideas or find them too difficult to grapple with so instead they have lengthy discussions about something they can understand - where to locate the bicycle shed.)

By contrast, a good steering committee can be a tremendous asset to a project.

What is the role of a member of the steering committee - what would you want them to do to help you, what would you want them *not* to do?

You would want them to:

- Help the project manager secure resources. So as PM you want people on the steering committee who are the ultimate owners of the bodies you will need for the project.
- Bang heads together within their own domains if their people are having difficulty making their minds up.

- Interact with other steering committee members to resolve inter-departmental disagreements.
- Make timely decisions.
- Publicly back the project manager's decisions.
- Support the project.

But, as suggested above, you would not want them to:

- Meddle in the detail.
- Keep changing their minds.
- Play politics (tricky to prevent!).
- Undermine the authority of the project manager.

Whose job is it to make sure the steering committee members understand their role? The project manager's. Book one-on-one meetings with each of the steering committee members and lay it on the line for them, make it very clear what their role is and what their role isn't. You, as project manager, might be a relatively junior person and you are now having to tell a senior manager or Director what their job is and what it isn't. This requires a bit of courage. It is not without reason that good project managers get paid a lot of money.

Does every project need a steering committee? Not really. If the whole effect of the project will be felt within the sponsor's empire he is almost by definition a one person steering committee. One can debate whether the project manager is on the steering committee or simply attends it. It doesn't matter. But the project manager clearly must be at steering committee meetings.

Role of Project Manager

What does the project manager do? Many a team member has muttered that question under their breath.

The list of project manager's responsibilities is either very long or very short. The short one is: the project manager is responsible for everything. Let's break that into some of its component parts:

Define and get agreement to roles. If halfway through a project it becomes clear the sponsor doesn't understand his role, whose fault is that? The project manager's. Defining roles means defining roles upwards as well as downwards. Particularly if it is the sponsor's first project, take half an hour to run through with the sponsor what he can do to help his project succeed. Note, his project, not yours. And as we said on the previous page, do the same with each member of the steering committee.

Some projects go nowhere because senior managers aren't clear what the project is really trying to achieve. If you suspect this is the case take them offsite for a day and get them to sort themselves out and hammer out a clear mission statement for the project. In one case that comes to mind a project had set out to be all things to all men. After several false starts a new project manager was brought in. He made the Directors clarify the project's goals and strip down the project scope, and only then did useful things start to get done.

Secure resources and fashion into a team. Getting promises from Directors - the steering committee - to provide people for the project is one thing. Getting the managers who directly own those people actually to release them is quite another thing. We will address this in a later chapter.

Having acquired the bodies, you will need to do some team building. This could just be a get together meeting. But if your team comprises people from IT, HR, Finance, Marketing, Logistics and Admin, and there has historically been antagonism between some of these groups consider something grander. A weekend canoeing and abseiling may cost a few thousand but this investment in team building could save you a lot of money and a lot of stress later in the project. Definitely worth considering for larger projects.

Plan the project. This covers a multitude of activities. The degree to which the project manager personally plans the project will depend upon its scale. If there are three people in the project team the project manager can easily plan their work. If there will be 300 people in the team the project manager will clearly need to delegate the detailed planning.

Design project control mechanisms. How will you report progress, control change, manage risks, etc? Every project is different, you will need to design control and reporting mechanisms that suit your project or at least adapt and modify any standard processes that are available within your company.

Manage project scope. If the scope is simply too big for the budget and timescale the project will fail. If you carve out a doable scope but then let it grow uncontrolled as you go along the project will fail. Managing project scope boils down to learning how to say the most important word in the project manager's vocabulary: "no". More on this in the next chapter.

Manage and report progress. This rather assumes there is progress to report, of course. We will cover tracking, controlling and reporting in a later chapter.

Be accountable for quality. The project manager's appraisal should depend not only upon meeting dates and budgets but also upon the quality of what is delivered. Even if the project manager is managing a project that is outsourced to a software supplier the project manager in the commissioning company should be held accountable for the quality of the software delivered by the software house. Sound unreasonable? Well, *someone* must protect the company from the potentially disastrous consequences of implementing a system that doesn't work. If not the project manager, who? After your company has gone bankrupt it's a bit late to sue the software supplier. See the chapter on quality management.

Reward and punish. Handing out the occasional small financial award to a team member who has excelled can really boost team morale, if they view it as well deserved. Where does the cash come from? The sponsor, of course. Or put another way, put a line item in the business case headed "team awards and rewards".

Characteristics of a Project Manager

What characteristics do good project managers share? As you would expect project managers are a pretty diverse bunch, but certain personal traits may help.

First and foremost they like managing projects. Managing projects is not something people are neutral about. They either like it or they don't. Why would anyone want to do a job in which you can all too obviously fail spectacularly and if you succeed people will shrug their shoulders and say you just did your job? Others relish the challenge and like the feeling of accomplishment, of getting something done that may not have happened without them.

Good project managers:

- manage rather than co-ordinate, preside or spectate
- are natural planners
- don't like surprises, so they plan thoroughly to try to prevent them
- are effective fire-fighters - when the inevitable surprises do occur they sort them out quickly and decisively
- reward and punish - not dealing with someone who isn't pulling their weight can destroy team morale
- are good motivators, good team builders
- address conflict rather than leaving things to fester
- do not hide in an office, they walk around and ideally sit physically in the middle of the team so they are approachable
- get consensus whenever possible but dictate when necessary

Project managers need all the personal skills that any manager needs so project managers should not only attend courses that teach them how to manage projects (may we plug this excellent project management course) they should also attend people management training: leadership skills, influencing skills, appraisal skills and so on.

Most of all, good project managers MANAGE. They do not just get swept along by the current (see The Tale of Three Project Managers). They grab the project by the scruff of the neck and manage it.

However, it is sometimes the case that project managers do not feel sufficiently empowered to manage and control the things/people they need to manage and control in order to be successful. This problem has a solution.

The sponsor should make it clear to everyone that the project manager is operating with the sponsor's full authority: "take issue with the project manager and you are taking issue with me; disagree with him and you are disagreeing with me; I don't care how senior you are, what the project manager says is what I say. Now if he's obviously got it completely wrong come and have a word with me, but other than that what he says is what I say."

Will the sponsor think of making some public pronouncement like that at the beginning of a project? Probably not. Whose job is to write the script for the sponsor? Yes, the project manager's.

How many project managers should a project have? One. That is, one reporting to the sponsor. The project manager may need others below him managing parts of the project or specialist teams, but there is only one project manager accountable directly to the project sponsor.

Project Manager Career Path

And who should this project manager be? An IT person, a business person, an external consultant? If the project is installing new servers with little or no effect on the users (one hopes) it may well be appropriate for an IT person to manage that project. But a project to develop a new software system that will run a key part of the company's business? That is a key business project and should be run by a business person. A significant proportion of the project team will be business users and they will do the most important tasks in the project: defining the business requirements and ensuring the proposed system design meets the business requirements and will be useable.

However, because process-based companies don't naturally breed project managers there may be nobody in the business with the requisite project management skills. This is an argument for nurturing project management skills in the business, not an argument for leaving it all to IT. Some companies, recognising that projects determine their future, are actively growing PM skills in the business. Training is obviously part of the answer. For non-critical projects assigning a business person to the role of project manager supported by a PM from IT aids skills transference. For a large project, bringing in a consultant project manager and assigning a business person as his deputy can also be an effective way of learning the project management ropes.

(Though beware some consultant project managers: bringing in a Californian PM who *knows* he can get the project done in 6 months when everyone else thinks it'll take a year can be dangerous. You may discover too late that his 6 months pre-supposed the project team would work 90 hours a week whenever he asked them to. Whilst that may have been true on his previous project in Silicon Valley, public sector employees in provincial England may feel more comfortable sticking to their regulation 35 hour week and give such a suggestion pretty short shrift.)

In larger process-based companies that nevertheless need to do an increasing number of projects to live long and prosper, it may be appropriate to have a small group of professional project managers reporting to a Director who acts as their career manager (the Change Director on the project organisation chart). These PMs could be ex-IT, ex-Finance, ex-consultants - but their passports now say "Project Manager" rather than "Accountant" or whatever. These '3 star' project managers would stand ready to manage the company's major projects. And if the project manager's line manager is a Director the project manager comes with a degree of ready made authority.

Two star PMs might be business line managers who from time to time manage medium sized projects, and one star PMs are business people just dipping their toes into the project management water. But we have a conscious project manager development programme within the business.

If projects determine the company's future it may be wise to have senior, competent, professional project managers managing those projects.

What's the difference between a project manager and a project director? About fifty thousand a year. Someone managing a really big project will often have the title project director to reflect the scale of the role. So our 3 star PMs may often take on the title project director when managing large projects.

Strictly speaking programmes never end - projects do. BMW might have a never ending (they hope) 3 Series programme. Within that there will be a project to develop the next version of the BMW 3 Series and even an overlapping project to start thinking about the 3 Series model after that. However, you will sometimes find large one-hit projects referred to as programmes. They may consist of many parallel projects (e.g. software development, hardware installation, office move, user training) which will all end at more or less the same time. The boss is called the Programme Director and may have many (sub) project managers reporting to him. Titles don't matter too much as long as the roles and responsibilities are clear.

Speaking of titles, the HR department in some organisations inhibits progression to a more project-friendly culture. In one example the HR department refused point blank to allow a person to take on the role title of project manager because he wasn't a Manager grade employee. And the idea of giving someone the title of project director would have been enough to cause apoplexy up there in HR's ivory tower. Also, in some parts of the public sector it can be difficult to pluck someone from the ranks, give them the temporary title of project manager and pay them a bonus if they bring the project home successfully. That upsets quite a few applecarts. All part of the process-based, project-inhibiting culture that we need to work to overcome.

Project Management qualifications are increasingly popular. However, if someone has attended a 5 day course and passed the end of course exam (Prince2, PMI, APM, ISEB, etc) that of itself tells you nothing about that person's ability to manage projects.

Have we covered everything a project manager needs to do? Not yet. You might say that the whole of this book is about what the project manager has to do or get someone else to do for him.

Role of Project Office

In a 6 person project the project manager can handle the project admin himself. In a 60 person project the PM will need a project office to handle the admin for him:

- acquire office space, order equipment
- set up and administer cross-project mechanisms such as change control and issue logging
- consolidate sub-project reports

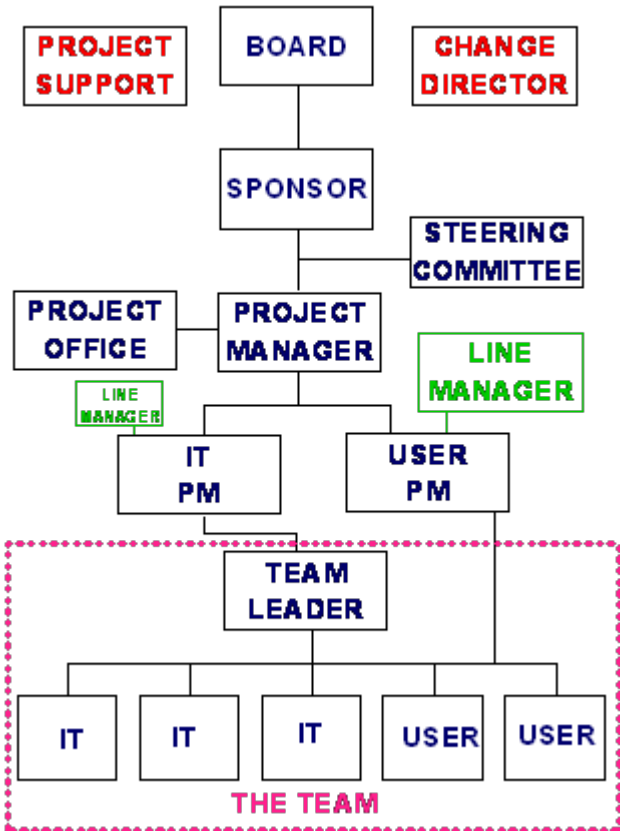
- financial tracking, adding up the numbers for the project manager
- project librarian
- etc.

Should the project office produce, own and update the project plan? No. This is a job for the team leaders as we shall see.

The project office provides admin support to the project manager, they do not manage the project for him. In a large project there can be literally thousands of documents and version of documents: without first class version control the project would soon descend into chaos. Want to know how many issues have been open for more than two weeks? Ask the project office. Or, if you want to get this information instantly on your laptop at any time of the day or night, ask the project office to set it up for you. A good project office is vital in a large project. Civilisation is underpinned by the competence of its administrators.

Role of Project IT Manager

If the overall project manager of an IT project has little or no IT knowledge they will need someone from IT to pull together the IT team and manage their activities and report their progress. This role is very similar to that of the overall project manager, if less broad in its scope.



Role of Project User Manager

Some would argue that the project user manager role is the most important role on a software project, the most difficult to perform and the role least likely to be properly defined or assigned. A pretty deadly combination. Furthermore some methodologies don't properly include this role.

However big the project there is only one project user manager. If it's time for a cup of coffee wander off and see if you can list, say, three specific responsibilities for this role. Glance at the organisation chart to see where the user PM sits in the hierarchy.

This is what the project user manager should do:

Represent all who will use the project's deliverables. Ensure that all users have a voice and their needs and views are considered by the project.

Find and commit business resources. It's all very well for the steering committee to make well-intentioned promises of boundless user people to work on the project. The project user manager has to (on behalf of the project manager) go out into the business highways and byways to prise out real, named, warm bodies to do all those things users need to do: define requirements, agree designs, user testing, etc.

Arbitrate amongst users. Users in Finance may want things done one way but users in Marketing think the requirement is quite different. The project user manager has to be empowered to resolve such disagreements so that a single view of the users' requirements can be given.

Sign off the requirements document. The project user manager signs off the requirements document which says to the sponsor: "your requirements have been fully defined and are complete and correct." And if they later prove not to be, the project user manager won't be getting much of a pay rise this year. Quite an accountability.

Sign off the User Functions Design document. At the end of the User Functions Design step, the user project manager signs off the UFD document: "Sponsor, the User Functions Design document is complete, correct, satisfies our business requirements and the system as designed is useable." And if it *isn't*... An awesome accountability.

Put yourself in that position: you are accountable for the correctness of the design for some huge new system that will eventually run your company's whole business. We are at the start of the User Functions Design step: we are building the plan for that UFD step. In two month's time, when the UFD document is finished, you must sign it off. Would you do nothing for the next two months, and when the document arrives on your desk shut your eyes and sign it then take it to the sponsor and say: "it's fine, you can sign this off, sponsor"? Don't think so.

As part of the planning of the design step you approach, let's say, the marketing manager and ask for one of his people to work full time for the next two months in the design team to ensure that the design meets the requirements Marketing have specified. But the marketing manager says although he'd like to help he cannot spare anyone for that. And you get the same brush off from every other business area you approach. What would you do (other than panic)? You would have to say to the project sponsor (directly or via the project manager): "Sorry, sponsor, I cannot perform that role. The role has no meaning unless it is me marshalling representatives from each business area and making them, and the people they represent, think through whether the design really does meet their needs and making them sign off on behalf of their domains. Only then could I sign off to warrant to you that the design really does meet your business needs."

The sponsor has just discovered he has a problem: he hasn't got adequate user involvement to ensure that the design will meet the business (i.e. his) requirements. If he (via the project manager) hadn't given someone this rather awesome accountability he might never have known - until it was too late.

In some methodologies this accountability is assigned to a member of the steering committee (project board) but that senior manager or Director might only devote a day or two a month to the project. We are talking here about a project user manager who works full time on the project to make sure the design is correct from the users' perspective. What we are suggesting is not at odds with these methodologies: it is, to be charitable, a way in which that steering committee member's responsibility may be delegated and meaningfully discharged. To be uncharitable it fills a fatal, gaping hole in some methodologies. Some methodologies' reluctance to assign such a significant responsibility to anyone below Director level may be borne out of a perception that it would be "unfair" to give such a responsibility to a more junior person. But the result can be in practice that *nobody* on the ground is truly accountable for the requirements and design being correct and - guess what - they won't be correct and the fun starts when the system goes live.

Authorise change. Having signed off the UFD document someone must authorise subsequent changes to it. As we will see in a later chapter the project user manager has a key role to play here.

Sell the project to the business. Explain to all affected parties what's coming and when. Set their expectations appropriately. On a big project this can be quite a big endeavour with road shows, executive briefings, demos and so on. But if, when the users get it, they all complain it's nothing like what they expected the project user manager has some explaining to do.

Manage acceptance. Acceptance Testing, if you insist. The project user manager signs off at the end of testing and says the system is fit to go live. If there is going to be a user acceptance testing step the project user manager might even project manage it.

Training and implementation. Listed last, but this starts in parallel with the IT Technical Design step and includes designing and building user training materials, developing the roll out plan and then near the end of the project executing the training and roll out plans. In really large projects, implementation, training and roll out, etc is such a large undertaking that it may well have its own dedicated (sub) project manager. Just another example of the infinite number of possible project organisation variations that the project manager must select from.

Now, if you're screaming at the page a) surely the project manager should do all of the things we have ascribed to the project user manager, or b) surely the team of super users, one from each user domain, should do all of those things it's probably because your experience has been of a) much smaller project or b) much larger projects than we are assuming for the purposes of illustration. (Though even on huge projects there should be one project user manager managing all those super users.) We will consider much smaller and much larger projects in a moment.

In summary, the project user manager is responsible for ensuring the project will meet the needs of the business.

Role of Team Leader

Our chart shows one team leader for simplicity. There could be many, both amongst the IT team and the business users. Typically they might look after a team of 6 to 8 people - maybe more, maybe less. What do they do?

Detailed planning. Produce the task-by-task, day-by-day plan for each of their team members and liaise with other team leaders to ensure teams' plans interlock. In large projects a member of the project office may facilitate this plan interlocking but beware, the plans must remain the property of the team leaders: if the plans are the project office's the team leaders may not be quite so committed to achieving them.

Control and report team's progress. The team leader manages and reports upon the work of his team. We will cover the mechanics of this in the chapter on tracking, controlling and reporting.

Ensure the quality of the team's output. In much the same way that the project manager will be held accountable for the quality of the project's outputs, the team leader will be held accountable for the quality of his team's outputs. Neither diminishes each team member's quality accountability as we shall see in the quality management chapter.

Technical leadership. IT team leaders usually provide technical leadership, guidance and coaching to more junior team members. User team leaders similarly may have greater business experience than their team members. A good team leader will develop the skills of his team members, support and encourage them.

Later chapters will cover the mechanics of estimating, risk management, planning, tracking and so on. Whereas in a small project the project manager will do all of this himself, in larger projects the team leaders do much of it, so in a sense the next several chapters explain many of the tasks a team leader would perform.

The role title project leader is sometimes used as an alternative to team leader (and even sometimes as an alternative to project manager). In large projects, beneath the project manager may be one or more project leaders and beneath them team leaders. And even sub team leaders below them. It just depends on the scale of the project. Apparently the project team that built the biggest of the great pyramids was around 13,000 strong (peaking at 40,000 - and you thought your project was big!) and had a project organisation that any project manager today would recognise: sponsor, project director, sub-project managers, team leaders, sub-team leaders and tightly knit teams doing the actual work. Project management is not a new invention. How *did* they manage it without Prince2 and MSP?

Team leaders manage the work of their team. Team leading is a good first step on the project management ladder. You will learn much more about managing projects by being a team leader in a large project than you will by managing a small project - you will get experience of many more control processes because many of those processes simply aren't needed in small projects.

We mentioned in the previous chapter the benefit of UFD design team members being present in requirements definition sessions and technical designers participating in the user functions design step. Ideally The UFD design team members will go on to become the team leaders in the technical design, build and test steps: this gives real continuity and carries the understanding of the users' requirements right through the project's life.

Role of Quality Leader

(Not shown on the chart). This is a member of the project team who ensures that appropriate quality checking activities are included in the project plan and that the team perform these tasks properly. This role will be defined in the quality management chapter as will the role's position in the organisation chart.

Role of Team Member

Even the lowliest team member has some part to play in the management of a project:

- help to estimate the work that will be assigned to them
- help to plan their own work
- sign up to their task plan
- report status of their work
- be accountable for the quality of their work
- alert their team leader of problems and issues etc
- suggest improvements to the project plan, control processes, etc...

The extent to which a team member can contribute to estimating and planning will obviously depend upon their experience, but the team leader (or project manager in a small project) should involve all team members in planning. This not only exploits their knowledge and experience but helps get their buy-in and commitment to the plan: it's their plan, not one imposed upon them from on high.

Role of Line Manager

This is not a project role yet line managers - the managers who own the people working on the project - can have a big influence on the project's chance of success.

Put negatively if a line manager has agreed that one of his people can work full time on the project for, say, the first two weeks of June but when June comes around he reneges and doesn't make the person available, that could cause major problems for the project. If June is eight months away from the project's delivery date it's hard for business managers who have little or no experience of projects to believe that this can possibly have any significant impact on the project. But non-availability of that person could have severe knock-on effects to the project schedule, and if that sort of thing happens repeatedly the project will be in trouble.

So, being positive, what should line managers do?

Commit resources. At the start of each project stage line managers commit, in writing, to release people - where possible named individuals - full time or part time at specified times. This implies thorough planning by the project manager and usually involves a good deal of negotiation.

Make people available as committed. Don't renege!

Manage careers. If a line manager makes someone available full time for six months to work on a project he nevertheless remains that person's line manager and must keep in contact with them. Ensure they don't feel cast out, assure them there is a job for them to go back to when the project ends. Manage their career.

Reward project work. When a person's project assignment finishes reward them for that project work. For example, if they spent half of their time over the past year on a project or projects, half their appraisal should depend upon how well they did that project work. But how does the line manager know how well his people have performed on projects? He asks the project manager.

This is a classic symptom of a project-based culture. A person has a career or line manager who manages them as a person but does not manage their work. He rents them out to projects and appraises them on feedback from project managers and others.

Think about this from the project manager's perspective. Everyone working on your project team knows that their next appraisal, and maybe pay rise, depends upon what you say about them to their line manager. Do you now have some power and influence over those people? You bet. Some of the project team members might even work for another company - you should have the same arrangement with their line managers.

This implies that at the start of a project stage the project manager must not only agree roles with everyone on the project but must also get their line managers to agree to base those people's appraisals on his feedback. This implies a lot of up-front work by the project manager. Which in a nutshell is one of the secrets of project success: a lot of up-front work

Other project roles

One could list a dozen or more further project roles: risk manager, test manager, project accountant, technical architect but since these are special cases of management roles and/or specialist team member roles we will not clutter up the works by listing them though we will allude to some of them subsequently. For example, the test manager gets a mention in the quality management chapter.

Each project is unique: it will require a unique, temporary project organisation. The roles will reflect both the project's needs and the skills and abilities of the people who will fill those roles.

A set of project management rules, if you have such a thing, will say which roles are mandatory (and the only mandatory roles are project sponsor and project manager) and would list the mandatory things each of these roles must do.

The project management guidelines would provide a fuller list of the sorts of things a sponsor and project manager might usually do. The guidelines would also list other possible project roles (project user manager, etc) with a suggested set of responsibilities. The roles and responsibilities that appear in these guidelines would be very much tailored to the sorts of projects your company does. The guidelines do not tell you how to organise your project, they provide a starting point, samples to help you work out what's appropriate for your project. You might decide that one person should take on the roles of project manager and project user manager - but if you do, that person in principle gets both lists of responsibilities to perform, not one or the other. You might want to invent a completely new role because of your project's unique needs. It's up to you if you're the project manager.

Never just assign titles. Saying: "will you be Project User Manager?" may convey nothing about the role to the listener. A good technique is to discuss with the person what you'd like them to do and then get them to write down their own role description (one side of A4 max) and replay it to you the next day. Then you know whether they've taken it on board or not. (Though you might not want to try this approach with the sponsor.)

One company had a template for their Project Definition Document. It helpfully listed their standard project roles - sponsor, senior customer, senior supplier, project manager, etc with a box beside each role for the name. And what happened? Project managers would write a name in each box and that was project roles taken care of. No thought about whether a role was *needed* or not, no discussion with the person concerned. Madness!

There will be disputes during the project. For example, if the users feel their legitimate requirements are being ignored by the project user manager they must have the right of appeal to the project manager and, if they're still not happy, to their member of the project steering committee. If they can't decide, the sponsor makes a decision (toss a coin if necessary). All escalations to the steering committee should be accompanied by a recommended resolution from the project manager. Escalation paths need to be clear at the outset.

It's a good idea to take a photo of every member of the project team and arrange them hierarchically on a wall (and/or intranet site) showing each person's title so that anyone can see at a glance who's who, who does what and who reports to whom. This can also engender a feeling of belonging, even pride in being part of the project team.

Very small projects, very large projects

In the smallest project there are two people on the project organisation chart: the sponsor and the person who will manage and do all the work.

Slightly bigger, and we have a sponsor, a project manager (who won't spend all his time project managing) and below him a team of two or three people.

Up again, and we have something approaching our example organisation chart but, say, with one person acting both as project manager and project user manager, no IT project manager and an IT team leader managing the IT activities.

Up again and we have something akin to our chart, with one or more team leaders, which might imply a total team size of twenty or so.

Up again and under the project manager we might have a project IT manager, project user manager, quality manager, test manager, architecture manager, procurement manager and others, all reporting to the project manager.

Then we get into projects that are really collections of projects. So the head man might be called project director and under him a project manager for the (sub) project to develop the new order handling system, a project manager for the (sub) project to develop the new invoicing system, a project manager for the hardware (sub) project, etc. And some of the (sub) projects might be in-house, some outsourced and some may be hybrids with both in-house (say users) and outsourced (say IT) elements - and a well staffed project office doing their best to make sure everything is well organised. Quite a big task for the project director to sort that all out before the project/programme begins.

Clearly, you can't just lift a universal organisation chart and set of roles from a book of standards and adopt it as is.

So:

- design a project organisation that matches your project's needs
- ensure everyone from sponsor on down has a clearly defined role with no gaps or overlaps
- talk to each person, agree their responsibilities, get them to write it down and check back with you
- get line managers to commit resources and agree to take your input into those people's appraisals
- never just assign titles!

How to spot a good project

Ask the sponsor who is in charge. Without hesitation he will say he is. If it all goes wrong his job is on the line; he is the ultimate can carrier.

Ask the same question of the project manager and he will say the buck stops with him - the sponsor is just a figurehead.

If the project manager is a business person, the project IT manager will scoff and say that actually he, the project IT manager, is really running the project - what does the PM know about IT anyway?

Speak to the project user manager and he will tell you he is the lynchpin, the key determinator of project success: if the requirements are wrong we're all wasting our time!

The team leaders will tell you that actually they are running the project - doing all the detailed planning and tracking and getting the team members to do the work. (And by the way they probably are running the project.)

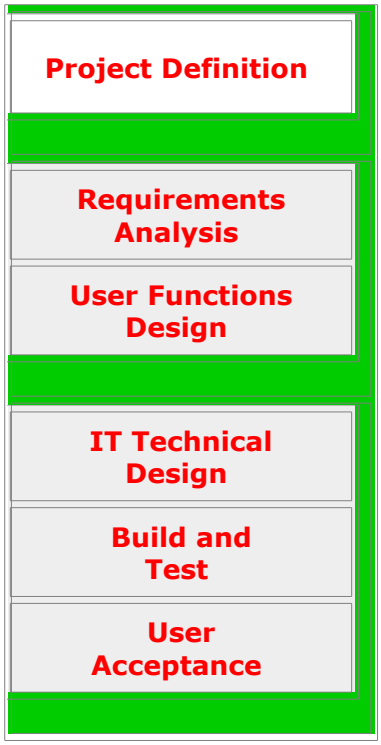
And what will the team members say? "We've no idea what all those people up there are doing - we're the only ones doing any real work, project success is obviously all down to us."

That is a very good sign. Everyone felt they were it, they were responsible, success depended upon them.

Contrast that with the opposite. Ask anyone who is responsible and they will say "not me!" and point to someone else and say "it's him". That's a worrying sign.

Whose job is it to make everyone feel responsible and accountable for project success? You've guessed it, our friend the project manager.

Chapter 4 - Project Definition



How long does it take to define a project? How long is a piece of string.

It could literally take five minutes to define a mini project. It could take a year or more to define a very large project.

Good project definition is key to project success. It is the foundation upon which the project will be built. If the foundations aren't solid the project may collapse.

How do we get from a bright idea or business need to a properly defined and planned project with an approved business case? We will describe the process under these 7 headings:

1. Definition stage plan and agreement: produce the plan for the project definition stage.
2. Evaluate the business need: is there a need for a project at all?
3. Find the best solution: don't just run with the first idea you thought of.
4. Investigate costs and benefits: build the financial business case.
5. Now define the project: roles, risks, resources, etc.
6. Estimate and plan the first stage: plan the first project stage in detail, later stages in outline.
7. Decide: go or no go.

We will describe these 7 steps as discrete and sequential - it will be obvious that in practice there will be a degree of overlap and iteration. To give some scale, we will discuss the definition for quite a large project - the project definition stage will take 10 weeks. At the end of this chapter we will see how everything scales down for smaller projects, but may of course be much more complicated for much larger projects. Remember that these seven steps are all within the Project Definition stage.

1. Definition stage plan and agreement

One morning you are driving to work and you have a fantastic idea. If your company had a system that could do x, y and z the company would make a fortune! What would you do when you arrived at the office? (No, not resign, the idea wasn't quite that good.)

You mention it to a few colleagues. They think it's a great idea. You work up a presentation and take it to a Director - someone you think would make an ideal project sponsor. He says: "This is a great idea I've just had, but I'd like you to research it fully: would it really make us money, what would it cost, could we do it...?" You have just been asked to do a full project definition.

It dawns on you that to research the need, devise a solution, build a business case, get a project planned out, etc would need the involvement of 25 people. You realise that project definition is actually going to be a project in its own right.

Being a natural project manager what do you do next? Little word beginning with "p": plan.

Planning is nothing to be frightened of. It simply means understanding what must be delivered and then working out who needs to do what and when to get those things delivered and signed off.

What must the project definition stage deliver? Three main things:

- Project Definition Document (aka PDD, PID, TOR, Project Charter...)
- Business Case
- Project Plan

So we work out who will need to do what to get us from where we are today - with just a rather vague idea - to having those things delivered and signed off and conclude it'll take about 10 weeks. As it happens, the 25 people who will be involved in doing the project definition are owned by 6 different line managers. From one manager you will need a person full time for the whole ten weeks, from another manager you will need two people for a week to build the financial business case, and so on. You negotiate with these six line managers for the bodies you need (aided by the sponsor if necessary) and invite them to confirm their agreement in writing (via a Stage Agreement which we will cover later).

And of course, if you were smart, you would have already had discussions with the line managers before you approached your prospective sponsor so that when he said "go get this potential project defined" you already knew there was a fighting chance you would be able to.

Having planned out the 10 week long project definition maybe you get the team together for a short meeting and have the sponsor say how excited he is about this idea and how he thinks it could be very beneficial to the company and to the futures of everyone in the company. The team think: 'wow, this is important'.

2. Evaluate business need

It's quite worrying how often this step is skipped altogether. People sometimes have great ideas about attractive *solutions* and nobody ever bothers to ask whether there is a problem that needs solving, whether there really is a business opportunity.

(In one case a project manager took over a large, troubled project half way through. After months of toil that was leading nowhere he was asked one day: "why is this project being done?" The project manager realised he didn't really know, so investigated. It transpired that eighteen months earlier someone had highlighted what they thought was going to be a problem with the ordering systems when new products were launched. A solution was devised that grew and grew and grew until it became a sizeable project, which then kept on growing when it was in progress. On investigation, the original problem turned out to be minor, to have a simple solution

and the project was cancelled. Ironically, with hindsight, one of the reasons the project had grown was that there *was* no need for it: had there been a need there would have been a date by which it was needed and a self-defining scope. Neither existed so the project had become "strategic". Several million dollars wasted. An interesting learning experience for all concerned.)

Do be sure you, as project manager, understand why the project is being done and that there is a real need for it. For some projects this means establishing not only what the problem is but also is it worth fixing it - what would it cost if we didn't fix it? What if we do nothing? For other projects, for example to develop a system to market a new product on the web: how many do we think we'll sell? For admin savings type projects, how many transactions would go through this new system and how much admin effort would that save? Will the project make us a fortune, save us a fortune or perhaps *must* it be done - for example for legal compliance? Techniques such as market research, high level business modelling, competitive analysis may be relevant. There are a thousand flavours of the question, but they all boil down to: does it look as though the project is worth pursuing.

If you're thinking: 'shouldn't the business have done all of this before the project starts?' it may be because you work in IT or a similar organisation and are used to the idea that projects start when they come to you ready defined (or more often than not when they are supposedly defined). We are following the project lifecycle right from the project's inception, from when it's just a gleam in the sponsor's eye. Even if you never get involved in project definition if you know what should be done during the project definition stage you can better judge whether the project you're being given has been properly defined or not.

Let us say that for our project we conclude the business opportunity is real.

3. Find best solution

Having established it looks like there's a real need or opportunity the temptation is to run with the solution you dreamed up that morning *en route* to work. But there might be a better way of meeting the need than the first one that happened to pop into someone's head.

Hold a brainstorm meeting (some prefer the term 'mind shower') with business and IT people. This of course will be one of the activities in the plan for your project definition stage for which line managers have committed people. Describe the 'problem' and then for an hour or so see how many solutions the team can come up with. Write down the solution ideas without critiquing them. Even an illegal idea might later spawn a really good (and legal) idea. It's not unusual to get a list of twenty or thirty solutions. Eliminate the whacky ones to arrive at a short list of solutions which could potentially fly.

For example:

- write a new bespoke software application
- buy a software package, use as is and tailor the business to fit the package
- enhance an existing system
- administer the new product manually using call centres in India
- outsource the administration to another company

These are radically different solutions that will have different timescales, costs, benefits, risks and strategic fit. We need to assess these solutions sufficiently to be able to choose one. A feasibility study may even be needed to see if some solutions would be technically feasible. Indeed, in some large, leading edge projects where feasibility is uncertain, a feasibility study may be the very first thing that is done. However, even if technical feasibility isn't in doubt, check for 'organisational feasibility': does your company have, or could it acquire, enough technically skilled people and enough appropriately skilled business people to take this project on in addition to everything else the company is already committed to doing.

(For projects that will buy and install a software package the choice may just be between several packages in which case a lot more detailed work may need to be done to match the company's requirement to the functionality of the packages before the choice can be made. If the packages on offer all have roughly the same cost, the broad business case can be built before the package is selected. And in fact package selection may be made at the end of the Requirements Analysis step. In that case, during the Requirements Analysis step the functionality of each competing package is compared to the company's requirement and the best fit wins. Competing packages often all seem to match the business need at the headline level and it's only when one gets down to the detail that the true fit can be judged. An example of a different project 'manufacturing process'. It is so important to be clear what manufacturing process your project will follow before the project starts.)

For the sake of illustration we will say that the solution that comes out top for our project is to write a new bespoke software application.

Bearing in mind we're only about half way through the project definition stage, is it too soon to be choosing the outline solution? It isn't. We must make the choice now otherwise we would not be able to do the next step.

4. Investigate costs and benefits

We can now start to build the cost benefit case. Clearly you can only estimate what the solution might cost when you know what it is.

The Finance department may have a list of typical major project cost areas and standard rates for converting people's hours into money:

- IT man hours during development
- user man hours during development
- implementation and roll out hours
- user training hours
- etc

The above non-cash expenditures, having been converted at relevant rates, can be added to cash outgoings such as

- travel
- hardware
- VAT (specially if your organisation can't reclaim it)

to give a total monetary cost. You may even co-opt a couple of people from Finance for a few days to help you build a project business case (and maybe things like monthly budget forecasts) that will meet the Finance Director's stringent requirements. Of course, the couple of people you co-opt are amongst the 25 whose managers committed them at the outset.

If this is an IT project, who is going to estimate the IT development cost? Clearly it has to be IT. It can take a lot of time and effort for the IT people to get sufficient information about what is proposed even to give a ballpark estimate. Again, these IT people are among the 25 whose availability you negotiated and secured in writing before project definition began.

Is there such a thing as a project benefit that cannot be quantified in financial terms? No. Every benefit can be quantified. Seeking a million dollars for a project that will 'improve staff morale' will guarantee you a quick exit from the Executive Suite. Seeking a million dollars for a project that will halve staff turnover and thus save two million dollars a year in training and other costs will probably get you a sympathetic hearing. You must at least quantify enough of the benefits to justify the project - any that you choose to leave unquantified may be icing on the cake. Naturally, in small projects the business case will be much less formal but the principle of cost justification should nevertheless be adhered to.

Only business people can assess the benefits this project will yield. It could take many people many days to get a quantified understanding of the likely benefits. Again, these people's time will have been committed at the start of the project definition stage. We begin to see that 25 wasn't such an outlandish number after all.

Let's recap. You had an idea, sold it to a Director who now thinks it was his idea and he asked you to do a full project definition which he will fund. You planned the definition stage (which actually you more or less did before you went to the Director), researched the business need/opportunity and found there really is a business opportunity, considered many ways of meeting that business opportunity and chose one. You then worked out the ballpark costs and benefits and the business case suggests a really good return on investment (ROI) and, given the ROI, you get the nod to carry on (or put another way, if the business case obviously didn't stack up you would stop at this point).

All of those steps are easy compared to the next step, because in the next step we have to decide what we're actually going to do.

5. Now define the project

Now and only now can we really begin to pin down the project that will need to be done in order to deliver the chosen solution.

Of course, one project definition stage could spawn a huge programme comprising a multitude of projects. Let's keep it simple: our project definition will result in one new system being developed. You now start hawking the proposed system around to potential users and they all get very excited: "what a great idea! And could the system also include reconciling physical stock with stock value in the accounts? And if we could handle international transfers we could use it in the Jersey branch too. And if it could do what we in Logistics have always wanted which is to..." And so the

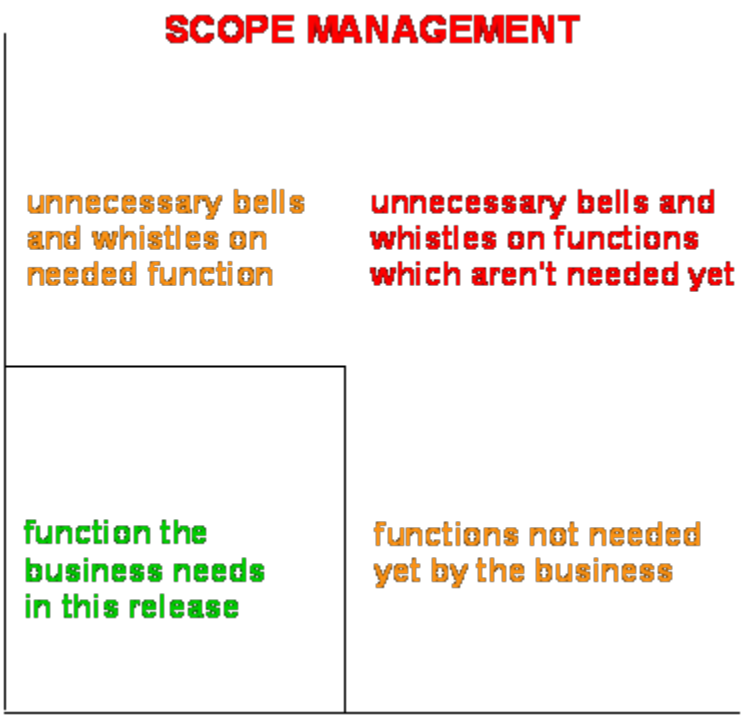
list goes on and on. And it dawns on you, the project manager, that if you tried to deliver everything that everyone would like in one go you'd be facing a big bang disaster project.

So you decide to break it into releases and you determine what will be included in Release 1. Why will that simple sounding step be a thousand times harder than anything we have done so far?

You must tell some very senior people that what they would like won't be in Release 1, it'll be in Release 3 - maybe. Will they say: "OK, that's fine"? Don't think so. They will escalate to the sponsor, threaten to withdraw support for the project, tell you you certainly aren't having any of their people for the project, brow beat and bully you, exaggerate the benefits of the things they would like included, claim their thing is a legal requirement, undermine, blackmail, bribe (if you're lucky), etc to get their pet things into that first release.

Will this be a pleasant way for the project manager to spend a few days? It certainly won't - these can be very bruising discussions. The temptation is obvious: avoid the question of scope altogether, leave everyone assuming that everything (including things they haven't even thought of yet) will be there on day one.

If it's a small project this will not be a problem - you can easily do everything in one go. If it's a large project you may have just secured project failure, or at the very least caused major grief during the project. This is where you can make or break large projects.



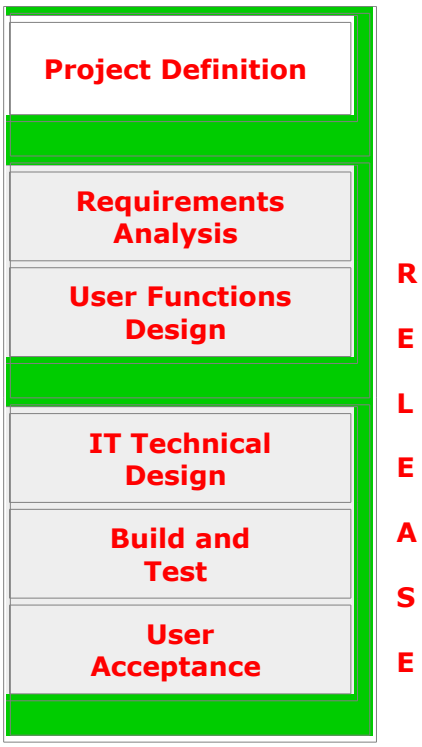
Take a tough line. Carve out a doable scope. In principle include in Release 1 only those things that are absolutely essential on day one, those things that constitute a useable system that can go live and at least run some part of the business.

Just possibly there is no useable sub-set, absolutely everything has to be there on day one and maybe it *has* to be done in one three year long big bang. If so, good luck. (And a pound to a penny says it ain't true - there's always some stuff you could leave till later.) But even here beware, for having concluded that all the headline functions must be there on day one, that may be construed as carte blanche to embellish those functions with quite unnecessary bells and whistles. Certainly, including in Release 1 unnecessary bells and whistles on

functions that aren't needed in Release 1 is the most heinous sin of all. It really is quite easy to make a software project four times bigger than it needs to be, as the chart on the left suggests.

Any mug can be everyone's friend at the beginning and say: "You want that included? Sure, I'll put it in for you." Good project managers know how to say "no": politely, firmly, rationally: "no".

If we survive that, we now have what we believe to be a sensible scope for Release 1. That outline business case we put together earlier may now need considerable rework: we must produce a business case specifically for Release 1, maybe with high level cost benefit guesses for subsequent releases.



Obviously, therefore, there will in practice often be considerable overlap and iteration between this step and the previous one, to the extent that the previous step may be completely subsumed into this step and a business case only produced after the Release 1 scope has been thrashed out. As always these things can be almost infinitely variable in practice (which is all part of the challenge that projects present).

Whether or not we produced a rough business case earlier, we now have a more precise cost benefit case for Release 1 that we can take to the sponsor. Having said: "*how* much?", he takes the business case to the Board of Directors who having said: "*how* much?" approve (we will assume) Release 1 funding in principle.

Now we can get down to defining the other aspects of the Release 1 project:

- decide who should be on the steering committee
- define and agree roles (at least for Stage 1 of Release 1, which we will say is going to be Requirements Analysis and User Functions Design)
- determine the development approach, the manufacturing process: RAD, Waterfall, Extreme Programming, package modification...
- determine how the project will be managed: issue handling, change control, status reporting...
- etc

Serious discussions about resourcing can begin.

We are now in week seven of our ten week project definition phase. It is all beginning to come together. We now hold a pre-planned, per-diarised Project Definition Workshop (PDW). The attendees will be:

- project sponsor
- steering committee members
- person who will manage the project (will not necessarily be the person who is managing the project definition stage)
- other key role players (project IT manager, project user manager, project office manager...)
- independent facilitator to run the meeting

The aim of the PDW is simple: to ensure all of the above have the same understanding of the project. The agenda should therefore include:

- project's goals and objectives
- business case summary
- scope and contents of Release 1 and outlook for future releases
- outline schedules
- staging and go no/go points
- roles and responsibilities
- who will supply what resources when
- how status will be reported, how change will be managed, etc
- risks and how they will be managed

Imagine that in spite of six weeks hard work prior to this PDW, at the meeting nobody can agree to anything. Is that a success or a failure? A bit of both maybe, but on balance a huge success: you may have just averted a major disaster. And it happens: you thought everyone understood and agreed, but they didn't. Sometimes you can have a one-on-one discussion with someone about their project role and they will agree it. But repeat exactly the same words to them in a public meeting like a PDW and they start to get cold feet and back away. Hold their feet to the fire, get them publicly to commit to their responsibilities and accountabilities. Given that all the key role players are present some adjustment of roles may take place during the meeting.

If the meeting does fall apart because there are major disagreements there will be a few days of urgent lobbying to do and even a second PDW to be held. (This is a good example of the difference between theoretical planning and real world planning. In theory, because of all the preparation, when the PDW is held everyone agrees to everything, so the plan shows one PDW. In reality people don't agree and further work and a further PDW is needed. A real world plan allows for this: it doesn't have to assume the worst case in every case, but equally should not assume everything will work out perfectly first time every time.)

What is in a Project Definition Document (PDD)? Fundamentally the things we listed as the PDW agenda. Indeed, the minutes of the PDW meeting could become the basis of the PDD. In principle a PDD documents what all the key players have already agreed to. A Project Definition Workshop (PDW) is an affective way of ensuring you have that agreement. Never write a PDD in secret and surprise the world with it.

Every methodology has its own take on what a PDD should be called (Project Initiation Document, Project Charter, Project Definition Report, Project Terms of Reference, even Business Case) and what it should contain but they all boil down to more or less the same things. See the Appendix for a suggested PDD table of contents.

One point worth making here: always have a project organisation hierarchy chart in the PDD. Don't just list the roles and role players. The hierarchy chart makes it clear who reports to whom. (It is not unknown for this chart to be omitted because it might upset someone to discover they were reporting to a person junior to them in job grade terms. But that's projects. If the project is to get done the project manager (and others) may well need to be have more senior people taking direction from them. Project-based companies take this for granted but it can cause real unease in process-based companies where there is a clear pecking order based on job grade.)

6. Estimate and plan first stage

There could be a number of (sub) projects all kicking off at the same time as a result of this project definition in which case each (sub) project manager would need to plan the first stage of their (sub) project. (Sub in brackets because some people would call them sub-projects, others would call them projects and maybe call the overall thing a programme.) But as we said we are keeping it simple by assuming that only one project will start now - Release 1 of our new system, and that Stage 1 of Release 1 will comprise the Requirements Analysis and Users Functions Design steps. We must now plan this stage in detail and get resource commitments.

Planning this stage is like planning any other stage: working out what must be delivered (in our case a requirements document, a UFD document, a plan for stage 2, a revised business case and probably several other things too) and then planning who must do what work and when in order to deliver those things.

Note that one of the outputs from any stage is a plan for the stage that follows it (assuming there is one).

A key part of the planning of Stage 1 will be negotiating with the managers of the people we will need - users to define requirement, a business analyst or two, IT designers to be involved in the requirements workshops, the user and IT people who will do the User Functions Design and IT technical designers who will participate in that, a project office person perhaps, someone in the latter part of the UFD step to start thinking about testing, etc, etc, etc.

We may have those promises of bodies from steering committee members and other Directors but once again we are now talking to immediate line managers. Having obtained verbal agreement we formalise them in writing in a Stage Agreement for Stage 1. An outline plan will also be produced for Stage 2 (in our case Stage 2 will comprise IT Technical Design, Build and Test and User Acceptance). Stage Agreements and the mechanics of planning and scheduling will be covered in later chapters.

Everything is now in place, the project has been DEFINED. The project's business objectives are clear. An overall business case indicates that a programme of many releases would be a good investment for the company. The cost benefit case for Release 1, and thus the budget for Release 1, has been approved by the Board of Directors. The project manager has defined and agreed all the roles such as steering committee members, project user manager, etc, and the role players have agreed to perform those roles and their line managers have agreed to appraise them on how well they perform those project roles. A PDW has been held and we have buy in from

all the key players. Stage 1 has been planned in detail and resources secured for it, the remainder of Release 1 has been planned in outline. We're ready to roll! But first we must get formal permission to begin.

7. Decide go or no go

This meeting should be a formality. The project manager presents to the sponsor evidence that everything has been defined and planned, risks have been assessed, resource owning managers have committed the necessary bodies, etc. In other words we know what we're doing and we're ready to start doing it.

The very fact that the project manager must make this presentation to the sponsor should be a powerful incentive to get the project properly defined and planned. You would not go to that meeting saying the objectives are not very clear, the scope isn't clear, you're not sure what it's going to cost and you haven't really got any troops lined up to do it. Another swift exit from the Executive Suite. This meeting is like a Summit Signing Ceremony for an International Treaty. There is no discussion or negotiation at the meeting, it's just a formality, but the fact the meeting is in everyone's diary focuses minds wonderfully and work gets done to hammer out the details prior to the meeting.

In our case the project manager confirms Release 1 will cost about $1M (which the sponsor already knows having taken the business case to the Board of Directors), and the project manager says he is confident that Stage 1 will cost $300K or very close thereto.

The sponsor congratulates the project manager for managing the project

definition so well and for getting everything sorted out and the sponsor gives the project manager written authority to spend $300K on Stage 1. As we shall see, at the end of Stage 1 the project manager will have another formal go no go meeting with the sponsor to persuade him to part with the funds for Stage 2, which he currently thinks will cost about $700K.

Smaller projects, larger projects

In our example project definition took 10 weeks, involved 25 people and produced several outputs: project definition document (PDD), business case, Stage 1 plan and agreement and an outline plan for the remainder of the project. And if we break the plan into all its component parts we might end up with quite a long list (risk management plan, communications/reporting plan, quality plan, etc - see later chapters).

However, if your project is very small, project definition might take a couple of days of informal discussion and your PDD, business case, project plan and stage agreement could be combined into a single document that covers two sides of A4.

We must scale the controls to match the need of the project so that we don't smother small projects in a blizzard of documents they don't need.

By contrast, the project definition stage for a very large project, particularly if it is to be outsourced to a third party, will need far more paperwork than we have contemplated: formal legal contracts, health and safety documentation, many project plans for the many (sub) projects, etc. Furthermore, if you are managing a large outsourced project, you will find your Procurement Department, Legal Department and others will be very heavily involved, and there will, quite rightly, be strict standards you must adhere to and processes you must follow (tendering procedures, etc). But if you are new to project management we would hope it's very unlikely you'll be given such a project to manage as your first project!

There's practically infinite variability in the size and nature of projects and thus in the size and nature of project definition stages. Before you start you've clearly got to work out what's the most appropriate way of defining your project: a few days discussion, a formal project stage much as we described...? Either way, there should at the end of the project definition stage be a formal meeting with the project sponsor at which he, in writing, gives the project manager authorisation to spend company cash on the project.

Does doing project definition properly mean you have unlimited time to do project definition? No it does not. Project definition should be like any other piece of project work: planned and managed so that it is done quickly and properly. And if you do project definition properly the project itself will take less time, cost less, have happier users and be less stressful.

Good project definition is a very good investment.

In some organisations, particularly those hidebound by project management methodologies, some people - we hesitate to call them project managers - hide behind the bureaucracy and use it as an excuse for inaction, inertia and

procrastination. "Sponsor, I can't do anything on that until you give me a project authorisation number." "Sponsor, I can't do any work on that until you give me a PDD." Rather say: "Yes boss, I'll get that done for you - I'll get a project number, work with you and others to get the project defined asap and a PDD produced then I'll get back to you with the costs and a project plan, etc and hopefully you'll decide to proceed."

In conclusion:

- projects have many causes: legislation, problems, company strategy, new technology
- there must be a project sponsor
- plan the project definition stage so it's done efficiently and effectively
- consider alternative solutions
- hold a project definition workshop to get agreement and buy in from key players
- always have a go/no go meeting at the end of project definition: make the sponsor decide go or no go.

Good project definition makes project success much more likely.

Chapter 5 - Risk Management

Here's a nice thought: you're going on holiday. But what might take the edge off your week of bliss?

- Forget tickets, money or passport
- Lost luggage
- Plane crash
- Lose passport, money or valuables while away
- Weather isn't what you wanted
- Food poisoning
- Fall out with travelling companions
- Get kidnapped
- Get injured
- Run out of money
- Work pressures force cancellation of the holiday
- Home is burgled while away

And we could all list a dozen more risks whose consequences range from minor inconvenience to major disaster.

Let's look at the first risk on the list above and consider how we might address it:

Forget tickets, money or passport

- have a list of things to take
- make a note of ticket numbers
- take a credit card
- keep a photograph of passport in your suitcase
- insurance

The list of things to take should help avoid the problem of forgetting something. But if in spite of this you still forget your passport, you've got a big problem - they won't let you on the plane. It's not very likely you'll forget your passport but if you do the impact will be high.

If your passport is stolen while you're getting crisped on the beach, that photocopy in your suitcase should get you back into the country without too much hassle. You haven't avoided the risk of having your passport stolen but you have mitigated the consequences.

If your camera is stolen you can claim on the insurance. Your camera has gone along with all your pictures but at least you've managed to transfer the financial loss to someone else.

Let's consider another of the risks and how we might address it:

Lost luggage

- well labelled bags
- half your clothes in your partner's suitcase
- essentials in hand baggage

If one suitcase goes missing at least you've got half your stuff. If all your luggage goes astray would that have a high impact on your holiday? It depends. If it's a skiing holiday a bit of a disaster. If it's a naturist holiday not so much of a problem.

Plane crash

- take your lucky rabbit's foot

The plane crash risk is very unlikely to happen though rather high impact if it does. In practice you'd probably accept/ignore this risk.

We've seen how some risks can be avoided, some mitigated, some transferred to others and some accepted/ignored.

Suppose you run a company and it will go out of business if you do not comply with new legislation. That is rather obviously a risk to your business. So you establish a project to do whatever is needed to comply with the legislation. The resolution of the business risk becomes the benefit side of the project business case. The project may be simple and easy to complete in the required time: we have a low risk project solving a high business risk. But if by contrast the project is large, complicated and very hard to do in the time available we have a high risk project attempting to solve a high business risk and the Board may want to give the project a lot of assistance.

Equally, a minor business problem could be addressed by a project that is high risk. There is no significant business risk but the project manager has a high risk of failure. Business risk and project risk are different things.

And there is another category. Imagine your project develops a new IT system but your project actually finishes just before implementation. There is clearly a risk that the system might not work, but is that a project risk or not? You could argue that it isn't a project risk since it happens after the project has ended. However, your project makes a commitment - whether explicitly or implicitly - that the project's product will work when it goes live. The project manager would be wise to consider failure to meet this commitment as part of the project risk assessment.

Project Risk

In this chapter we are primarily considering project risk, that is things that might cause project failure:

- things that might prevent us coming in within the budget we are thinking to committing to

- things that might cause us to fail to meet any dates we are thinking of committing to
- things that might cause us to fail to meet any other commitments we are thinking of making

Though "risks" can also go the other way: that techie you're going to hire may be twice as good as expected and the "risk" is that he does the work in half the planned time. Some say these "risks" (better called opportunities) should be assessed and monitored in much the same way as negative risks, to make it more likely they are exploited.

All of us have been on holiday many times so we all have in-built mental holiday risk checklists. But imagine you are asked to manage a project and you've never managed anything similar before. You'd have no idea what the pitfalls might be nor what to do about them. If you had a checklist, like the holiday risk checklist, that listed the things that had caused problems in previous similar projects, and what had been done to address those problems, that could be very valuable to you and help you deal with similar problems in your project.

Let us see how we might manage risks in projects. We will describe the process under these 5 headings:

1. Measure - realise what risks you face, assess their probability and impact.
2. Minimise - identify what could be done to remove or reduce risks and do it.
3. Mention - speak to the sponsor, don't keep risks secret.
4. Monitor - keep risks under review, monitor and reduce them as you go along.
5. Modify - share your experience, update your company's risk checklists.

You will notice that the initials form an easily remembered acronym: mmmmm.

1. Measure

If your project involves HR, Marketing, Finance and IT, only someone from those divisions can really tell you the risks their division poses to the project. Will they really be able to supply the promised resources? Will they really empower somebody to represent them and make decisions on their behalf? And only IT specialists can tell you how risky the proposed technology is.

Hold a half day risk identification meeting with representatives from all involved areas plus experts who you think will be able to help identify risks. Initially do not use a checklist. Ask everyone to articulate what they think could cause problems in the project - tell them beforehand you're going to do this so they give it some thought. List everything everyone mentions. Then - if your organisation has one - go though the risk checklist: that will probably highlight risks you hadn't thought of. Don't do it the other way round, don't go through the checklist first as that can limit people's thinking. (See the Appendices for an example of a risk assessment checklist.)

You now have a list of things that you think could cause problems for your project, i.e. risks. For each risk assess how likely is it to happen and how severe its impact would be. Avoid esoteric discussions on probability theory and percentages. Keep it

simple. Beside each risk write High, Medium or Low under these two headings: probability and impact. For some risks you might decide you need an 'Unacceptably High' ranking.

Then re-order the list to show high probability high impact risks first - clearly these are the ones that represent the biggest threat to your success. Perhaps during the meeting you can devise ways of removing or reducing some of those risks. Where you can't, you might decide to assign risks to individuals to follow up.

Tools that employ risk weighting and calculate an overall risk score in an apparently scientific way may add value, but can obscure the fact that one single risk, which may not even be considered in the tool, can make the whole project a complete non-starter. Whatever techniques you use to assess, measure and understand the risks - brainstorming, checklists, tools - remember that the aim is simply to focus the attention of those involved in running the project on those things that are most likely to cause grief or even failure and ensure they do something about them. Complicated and mathematically elegant processes can cause one to lose sight of this. Similarly, don't clutter up the works with hundreds of things that are unlikely to happen and would have minor impact even if they did. List them (if you really must) but otherwise ignore them.

For very small projects risk assessment might involve getting the team together a week or so before you are due to make you commitments to focus their mind on risks. A half hour meeting in which you ask the team what they think are the risks and maybe to go through a simple checklist (see the Appendix) may be all that's needed. Though risk reduction may still be difficult.

For very large projects you may have much more formal processes, perhaps formal health and safety risk assessments, and you may even employ consultants to assist if the company has no experience of projects of that size and nature.

2. Minimise

We now have a list of things we think will threaten our project's success and we have ranked them with highest likelihood/impact first. It may only have taken half a day to do this - to realise what risks the project faces - but identifying what can be done about those risks, and doing it, could require some heavyweight, time consuming, toe-to-toe negotiation.

If the project scope is simply too large to fit into the desired timescale, some forthright discussions with the sponsor may be needed. If one of the major risks was, say, that HR may be unable to provide a person properly to define HR's requirements, a meeting with the HR manager may be appropriate. If a firm commitment is obtained for a knowledgeable HR person to be definitely available and empowered to represent HR you may conclude the risk has been removed. If you can only extract a half-hearted resource promise you may have succeeded in reducing the risk a bit but not in removing it. How big the remaining risk is depends, partly, on how important HR's requirements are in the grand scheme of things and to what extent you believe the promise. You have to make a judgement.

Some risks might be avoided altogether: you simply won't use that untried technology. For other risks, you will be able to reduce the likelihood of them

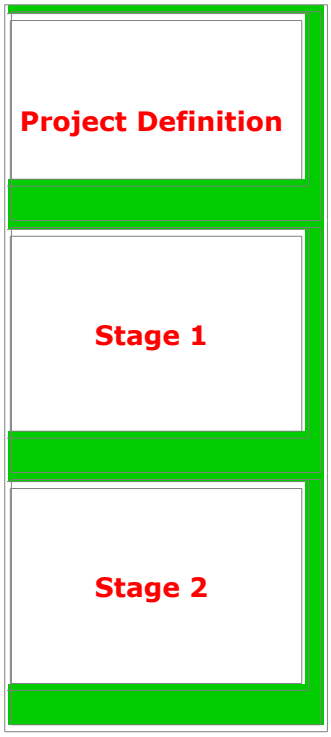
occurring. If you thought a couple of people might leave the project at some point (which could cause real problems) what would you do?

For contractors you might offer to pay them a bonus if they stay until the end of the project. They might still leave half way through but it's less likely.

For regular staff you might line up two backfills who would come in and replace them. In the plan you'd then have two tasks (not yet assigned to anyone), let's say each one week long, labelled 'hand over to your replacement'. As soon as you know someone is going to leave that becomes the last task in their plan. And of course the first task for each of the backfills would be a one week 'be handed over to' task. You haven't altered the risk of people leaving but you have potentially reduced its impact. Now, if you get lucky during the project, guess what? Nobody leaves and you've got 4 weeks of effort in hand! If you get unlucky, guess what? More than two people leave. This illustrates two things quite nicely: for some risks you put specific tasks into the plan to help mitigate the risk (the handover tasks) and it also shows it is all ultimately down the your judgement. Your judgement was that two would leave so you planned on that basis. You win some you lose some but do that across all the risks and if you're lucky it will more or less balance out.

For some risks there may be no specific tasks that can be include in the plan, so we simply include contingency in the budget (i.e. money) and in the schedule (i.e. elapsed time). One risk that exists on every project is that tasks are overlooked and not included in the project plan. When, half way through, you discover work must be done that is not in the plan what do you do? Raid the contingency. And you will never foresee all of the risks: things that aren't foreseen are sometimes called unknown unknowns - include contingency for them.

How much contingency should the budget and timescale include? It depends. A small, simple, familiar project may only have 5% budget and time contingency. A huge, complex, leading edge project may have 30% contingency, which means you're saying this: "I've planned work tasks for everything I can see we will need to do, but this is such new territory I *know* we will end up doing a lot of other work as well - I just can't foresee specifically what those work tasks will be, so I've allowed time and money for them under the heading contingency." Thirty percent contingency will be hard to sell. Another reason PMs get well paid.



Replanning of various sorts is usually what risk reduction boils down to: extra people, moving the end date, building in third party reviews, building in contingency, etc. Estimating and planning contingency will be further explored in the next two chapters.

When are we doing this risk assessment? As part of the planning of each project stage. So, as we near the end of the project definition stage we will assess the project's risks. We may even do two risk assessments at that time: one in great detail for Stage 1 and another for any later

project stages. Then, as we near the end of Stage 1 we will, as part of the planning of Stage 2, do a detailed risk assessment for Stage 2. Risk assessment is a key input to the go no go decision at the start of each project stage.

3. Mention

We have done what we can to deal with the risks: removed them, reduced their likelihood, included things in the plan to reduce their impact, put contingency in the plan, added extra people, adjusted budgets and dates... all to try and make it more likely we'll succeed. But we certainly haven't magiced all the risks away: risks remain.

Who owns the project risk, who ultimately is taking the risk? The project sponsor. But most sponsors, particularly of technical projects, have no idea what the risks are. The project manager has a duty to explain the risks to the sponsor before the sponsor gives the go-ahead.

But beware - there are good ways and bad ways of presenting risks to the sponsor. Do not go to the sponsor with a hundred PowerPoint slides listing several hundred risks most of which are minor. Instead try something like this: "Sponsor, we have been through the following process to assess the project's risks - workshops, checklists... Initially, sponsor these twelve things came out as very high risks. However, we have already taken these actions... and as a result eight of those twelve risks have been removed or significantly reduced." That is a very positive thing to say and builds real credibility for the next bit where you say: "But these four high risks remain." You must then explain the probability and impact of each. If you were the sponsor and the project manager said: "This risk is almost certain to happen and when it does your $5M project will be a total write off," would you give that project the go ahead? Probably not - unless the benefits are measured in billions and it's worth the gamble.

For example. "Sponsor, this risk remains: it is very probable that HR will not be able to supply resources to define their requirements. The impact will be that we will have to more or less guess what they want and in the worst case when we go live the system won't do what HR needs and will be unusable." Now, like any good project manager you're probably already thinking of ten ways that risk could be reduced! But assuming all those things have been tried and there really *is* no way HR can do it the sponsor may decide he will not start the project until HR can make the requisite resource available.

But if we said: "Sponsor, this risk will be very hard to contain and if it happens we could easily be a month late." The sponsor might say: "OK, try and bring it in on track, but I recognise the end date is a bit exposed."

You must mention the risks to the sponsor before he gives the go-ahead. Imagine you keep the risks secret and when the project is all going wrong you say: "well of course, sponsor, I knew all along it was high risk and bound to go wrong," you're going to get shot, and quite rightly so.

A face to face presentation of the risks should be mandatory for any significant project.

4. Monitor

We've assessed the risks, reduced them where possible and told the sponsor how great the residual risk is, i.e. how big a risk he is running by deciding to do the project.

And the sponsor has said: "I hear what you say, I understand all those risks but succeed in spite of them." At this point don't go off in a huff, put in place things that will make sure the risks are managed down as you go though the project. Don't just log the risks and forget them.

Don't try and manage hundreds of minor risks. Focus on those most likely to cause significant problems. Assign each of these risks to someone in the team: it's their job to ensure the team does whatever is needed to reduce that risk as we go along. For example, if we will need a technical specialist in 3 months but they are hard to come by, one way of reducing the risk might be to ring the agency every fortnight to remind them of our requirement.

In weekly team meetings risk owners report briefly on the status of the risks they are looking after: what is being done, what needs to be done to keep the risk at bay. These actions could be pre-planned or involve the use of contingency time and money. In a smallish project the team members would report risk status direct to the project manager in the weekly meeting. In a larger project the reporting might be to the team leaders in their weekly team meeting and the team leaders report on the more significant risks to the project manager.

Perhaps in our example the team leader reports there are now plenty of those technical specialists available. Or he might report there is no prospect of finding someone in which case you might decide to trawl the world for that expertise and prepare to spend a slice of your contingency to pay for it.

Once a month the project manager tells the sponsor the status of key project risks. More on reporting in a later chapter.

In very large projects with many significant risks it may be appropriate to employ a risk manager who does nothing other than help the project manager manage the risks.

Of course, new risks can crawl out of the woodwork as the project progresses - if they are significant add them to the risk register.

5. Modify

Some organisations have a Project Support function (more later). At the end of each project stage they ask the project manager what problems they encountered that they hadn't foreseen. And when someone trips over a risk that isn't on the organisation's risk checklist what do project support do? Add it to the checklist so the checklists evolve to reflect risks actually being encountered. Checklists because there could be several: hardware projects have different risks to software projects, so there could be a risk checklist for each type of project the organisation undertakes.

And if project support are smart they'll also ask: how did you deal with the problems or, with hindsight, how would you deal with similar problems next time? Project support collect these successful ways of dealing with risks and produce something like the holiday risk checklist - what might go wrong and what might avoid or mitigate those risks. This means that new project managers don't all have to learn about risks the hard way.

Monte Carlo

People have tried to devise elaborate mathematical risk assessment procedures. You answer a thousand questions, the tool whirs and clicks and comes out with a risk score - 63.724%. This can be very misleading: the scoring can obscure the fact that one risk actually makes the project a complete non-starter (and that specific risk may not even be considered in the tool). With checklists and tools there is a tendency just to consider the risks that happen to be listed. They also encourage averaging: you have a list of 20 risks, 6 high, 8 medium, 6 low - what's the overall risk? The temptation is to say medium. But again, one of those high risks could make the whole project too high risk to contemplate. Don't arrive at the overall risk by averaging. Projects are all different, no checklist will ever have on it all the risks your project faces. Risk assessment is a brain-in-gear activity, it is not about filling in a checklist and filing it away.

Some projects and manufacturing processes lend themselves to techniques such as EMV and Monte Carlo simulation. With EMV, the probability of a risk happening is multiplied by the cost if it does happen and the resulting amount of money (expected monetary value) put into the budget. So, 10% chance of a $1M problem biting, £100K goes into the budget. If you had ten such risks you'd have a £1M contingency pot. The chances are one risk will happen and nine won't and the $1M covers the one that happens.

Monte Carlo works like this: each task in the project is given a probability of happening anything between, say, 50% early and 200% late. A computer simulates the execution of the project many times, each time using statistical techniques for deciding how on time or otherwise each task will be. Sometimes the simulation will result in the project finishing early, sometimes on time, sometimes late. If, say, the most common outcome was the project finishing around 20% late that might be where you'd want to put your money. This is all fine in theory but it assumes you can give accurate values to all those probabilities. Keep it simple.

To summarise the risk management steps:

- Realise what risks you face, assess their probability and impact.
- Identify what could be done to remove or reduce risks and do it.
- Speak to the sponsor, don't keep risks secret.
- Keep risks under review, monitor and reduce them as you go along.
- Share your experience, update your company's risk checklists.

Or put another way: Measure, Minimise, Mention, Monitor and Modify.

Finally, managing risk is *not* the same as being risk averse. Jumping out of aeroplanes is a pretty high risk undertaking and you may decide it's wise to reduce the risk by wearing a parachute. Sometimes doing phenomenally high risk projects is

the right thing to do. But if it is very high risk everyone knows that and exceptional action is taken to make the project succeed in spite of all the risks. For illustration, in a very high risk, business critical project you might ask for the mobile phone numbers of every member of the project steering committee - who may in fact be the Board of Director - and have an understanding that if anything arises day or night or weekend they can help with, you will call them and they will do whatever it takes. And would you, the project manager, let everyone know you had those mobile numbers? The unspoken promise is: get in my way and your Director will be having a quiet word with you...

The Appendix contains two examples of risk assessment checklists: one for very small projects with about a dozen items on it and one for larger projects which runs to several pages. Even the long checklist won't have anything like all the risks on it that any given project will face - risks will be very dependent upon the project's nature and the envirnoment in which it is being done.

Chapter 6 – Estimating

Estimating is like getting rich: everyone wants an effortless way of doing it. Failing that, estimating can be done quite easily using sophisticated tools such as the one pictured.

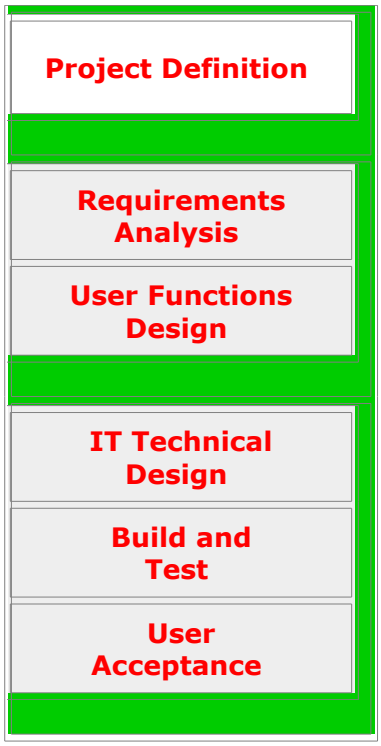
Look up from the page at the wall in front of you. How long would it take you to paint the wall? What's your estimate? Twenty minutes? A couple of hours? You've got the job.

Because our 'paint the wall' task of course included:

- agreeing the colour with your partner (a 2 month task in its own right?)
- buying the paint from a boutique paint factory in Faro, Portugal
- preparing the surface
- tidying up

So our two hour job is actually more like a two or three month job.

This happens, usually by the coffee machine: "I've got to go to a meeting, can you give me a rough idea how long it would take to...?" and under pressure you say: "about two hours?" But when you find out what's really involved it is more like 3 months work. So the first message from this chapter is: don't guess. We will see later how we might resist the pressure to generate random numbers.



What's the worst way to estimate how much a project will cost? Get one junior person who has never estimated before to have a guess and then we all become committed to that guess. We'll see later how we might bring experience to bear on this difficult task of estimating project cost.

Suppose we are at the start of a year long project, can we estimate its cost accurately? Almost certainly not, there are so many unknowns. Until we have defined the detailed requirements and done the design it's very hard to say what the build might cost. When we're in the project definition stage we may conclude we can only estimate with any kind of accuracy Stage 1: Requirements and User Functions Design.

So, we will do a detailed, task by task, bottom up estimate for Stage 1 and a much softer, top down estimate for Stage 2. When we near the end of Stage 1 and have the design done we will then estimate Stage 2 task by task, bottom up, and provide an accurate estimate for it (we hope!).

Top down estimating is getting as good an idea as we can of total project cost and bottom up estimating is getting a precise estimate for the project stage we are about to start.

Let us first consider some techniques for improving the accuracy of top down estimates.

Good project definition

We were miles out with the wall painting project estimate because we didn't understand the project scope - we did not have an agreed project definition. Crystal clear project scope is an obvious prerequisite to good top down estimating.

Include all costs

Ask someone from IT what a project might cost and they will think about the programming cost and - if you're lucky - the testing cost, add the two together and give the result as the project cost. But in reality that is just the tip of the cost iceberg. Even if the estimates for the programming and testing costs are spot on, it's what's left out that makes the overall estimate way out.

How might we overcome that problem? Have a checklist of all possible costs a project might incur: project definition, requirements analysis, design, documentation, team meetings, change, project management, contingency, travel, plus a very long list of et ceteras. This makes it much less likely cost items will be overlooked.

A good checklist includes peripheral costs too: it's all very well quoting $50K to develop a little new system but if three other systems must be modified to interface to it, those costs have to be included somewhere. And suppose a data conversion would be needed to get old data into a form usable by our little new system - was that cost in the $50K project estimate? A good checklist makes you think very broadly about all of the costs associated with the project, not just the narrow development costs.

These checklists are relatively easy to develop by looking at where the time and money went on previous projects. If that information isn't available, sit down with two other people for an hour to draw up such a checklist: you will be amazed at what you end up with on that list that you would have overlooked when doing an estimate.

When you have such a checklist, try this. Ask someone for an off the top of the head estimate for a little project. Then give them the checklist and come back ten minutes later for their revised estimate: you will be surprised at the difference.

Break it down

Never accept a single global figure as the estimate for anything. Always get it broken down by the major steps the project will go through.

If someone came to you with a breakdown for a new-build software project and said the IT effort will be as follows, would you believe the estimate?

Work with users to define requirements	8 man months
Develop the User Functions Design	8 man months
IT Technical design	2 man months
Build and Test	2 man months

Total cost 20 man months of IT effort. 18 man months to get to the end of the design steps then 2 man months to do all the programming and testing for a new build software project - does that ratio sound right? If it cost 18mm to do the requirements and design what do you reckon the build and test might cost - in your experience, what percentage of the IT effort is spent in the build and test steps of a new build software project?

Suppose in past new build software projects the IT effort had in fact typically been distributed as follows:

Requirements	15%
User Functions Design	15%
IT Technical Design	10%
Build and Unit Test	40%
System and User Test	20%

Would you now believe the 20mm estimate in the example above? At the very least you would need a lot of convincing why this time the work distribution will be so dramatically different.

Your company might have a number of such models - yardsticks against which to compare future estimates. For example, in package modification projects the IT work distribution might be more like this:

Requirements	20%
User Functions Design	20%
IT Technical Design	15%
Build and Unit Test	15%
System and User Test	30%

Most types of projects go through a number of steps such as these. As a matter of principle, never accept a single global number, always get it broken down by the relevant steps the project will go through and compare to past experience as a reasonableness check. Benchmarks such as this only work, however, if projects apply a consistent development (manufacturing) process: if some projects do properly detailed and complete user functions design but other projects do UFD at a much vaguer, higher level the effort breakdown would obviously differ from project to project.

Here's a quiz for you.

At the start of the project pictured on the right, the project manager said the total cost would be 1000 man days: 400 man days in Stage 1 and 600 man days in Stage 2. He knows that in the past, similar projects have expended 40% of their effort in Stage 1 and 60% in Stage 2.

Stage 1 has just finished. It actually cost 800 - twice as much as estimated. So, the original estimate for the project was 1000, 800 has been spent so far - what would you say the remaining project cost will be? The most likely answer is 1200 man days. But what would some project managers say? The remaining cost will be 200: 1000 budget, 800 spent, remaining cost 200. There is a rather big difference between 1200 and 200! Always look at what has actually been spent so far and using past ratios extrapolate forwards to get a more realistic picture of what the remainder of the project is likely to cost.

Rules of thumb

Imagine this. Your developers estimate a project at 49 man months of effort, i.e. one person could do it in 49 months (about 4 years) or it would take 2 people 24.5 months (2 years) or 4 people a year, etc.

Next day you tell the sponsor it's 49 man months of effort and whatever that is in money. He says he can afford it but wants the project finished in 3 months starting from... NOW. Can you do it? Bear in mind the project hasn't started so by definition none of the team is there yet. You say you're not sure so the sponsor offers you unlimited funds and access to IT consultants - whatever you need. Can you do it? While you are hesitating the sponsor points out that in the business they handle 1000 orders a day with 10 order handlers and if they got 2000 orders a day they'd use 20 order handlers and still do it in a day. So, the sponsor says, why don't you get in 49 IT contractors in on Monday morning and do the project in a month? When you look incredulous he says: "OK, tell you what: do it in 4 months, OK?" And if you have no rationale for rebutting his suggestion you may find 4 months is now your 'commitment'.

This is another difference between process-based thinking and project-based thinking. For operational activities you can simply increase the number of people - almost infinitely - as the people are operating independently. But in a project adding too many people just causes chaos, and we all know we can't suddenly employ a huge number tomorrow - they'd have nothing to do. In a project we have to start off with a few, build up and then come down to zero at the end.

So how long would a 49 man month project take? There is a rule of thumb known as the square root rule. Express the project effort in man months, 49, and take the square root of that number. You now know why we chose 49. That would suggest an elapsed time of about 7 months. This works well as a rough guide to the minimum elapsed time of IT development projects. Could we do the project in 6 months? Maybe, if we ban holidays and work weekends. But there is a limit. And if the project spans August it may take 8 months under normal conditions because of holidays. Obviously if you choose not to use as many people as you could, the project will take longer than the rule of thumb would suggest.

If everyone knows the square root rule, including project sponsors, as soon as you say it's 100 man months they know it's going to take around 10 months to get it done. This can result in fewer projects trying to do the impossible. Of course, if your team is happy to bring their camp beds into the office and work 20 hours a day you may be able substantially to break the square root rule. But would your team do that?

The square root rule is sometimes known as the golden square rule: the average team size is roughly equal to the elapsed time in months. Seven month project? Average team size of seven people. Some sources say that 7 should be the *maximum* team size, but we would suggest that on a 7 month project you could safely peak at ten or even a dozen team members without them tripping over each other.

In a programme of several sub-projects you'd apply the square root rule to the largest sub-project to get a rough idea of the programme's duration.

This is not a magic formula that tells you exactly how long a project will take - many factors could affect a project's duration. It is just a rough guide to the shortest duration it might be possible to do a project in.

User effort

In your experience, who estimates how many hours of user effort will be required for "IT" projects? If we are honest the answer sometimes is that nobody does. If you're the HR manager and nobody tells you how much work will be needed from your people what will you assume in your planning? A big round zero. Anything is better than a default assumption of zero.

Suppose that on past projects the relationship between IT effort and user effort had been as follows:

	IT Effort	User Effort
Requirements	1	1
User Functions Design	2	1
IT Technical Design	3	1
Build and Unit Test	4	1
System and User Test	2	1

So, in the requirements step for every man day of IT effort there had been a man day of user effort. In the user functions design step for every 2 man days of IT effort there had been 1 man day of user effort. Having estimated the IT effort per step you could at least get a very rough idea of the user effort per step. If you can estimate the user time more accurately you should, but anything is better than a zero estimate of user time being assumed because no other number has been mentioned.

The above are illustrations of how the essence can be distilled from past projects to yield rules of thumb that will at least guide project managers to the right ballpark when doing top down estimating.

Realism

In your experience do projects end up costing less than the initial estimate or more than the initial estimate? More? Why? Why do people tend to underestimate at the beginning? Because they e.g.:

- want the project to fly
- don't allow for change
- don't allow for management activities
- don't include quality checking activities
- don't include contingency

Why do project managers sometimes leave these things out? Guilt.

Suppose that on a past project effort was actually expended as follows:

Change	15%
Planning, controlling, supervising	10%
Quality checking	15%
Unplanned tasks	10%

It is only reasonable to assume similar expenditures on the next similar project and therefore to include these things in the next project's estimate. But please add those percentages up - what do they come to? About 50%. You must now go to your sponsor or Board of Directors with your estimate breakdown, 50% of which is change, quality, contingency... At this point even the most experienced project managers begin to feel guilty that they are padding the estimate and what do they do? Shut their eyes hope those things will go away and leave them out of the estimate. If you leave all of those things out the project is going to cost roughly double your estimate.

How might we make project managers be realistic and include the appropriate factors in their estimates? Make them personally accountable for the estimate they give at the start of each stage. At the end of the stage independently measure the actual cost and the closer it is to the estimate the bigger the project manager's next pay rise - and we may even pay him a bonus. And if the actual is miles away from the estimate

In this rather hard-edged culture would you include the relevant percentages in your estimates? You would. (Incidentally, the percentages for things like change and contingency will usually be lower for small, simple projects - but could well be higher for large, complex projects).

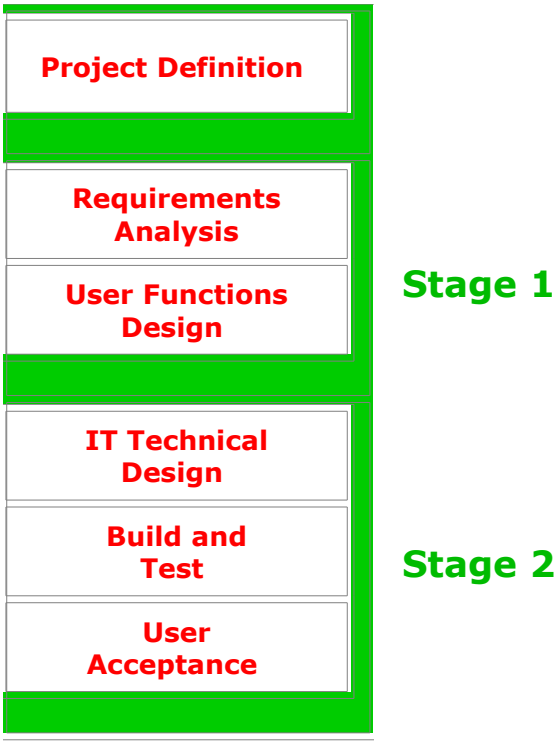
So, you do a professional job of estimating and include the appropriate percentages. You then present your cost breakdown to the sponsor, even the Board of Directors. They are horrified at the cost and tell you to cut your estimate. What would you do? Offer to reduce project scope perhaps. But no, they want full scope, just a lower estimate.

You justify your figures by showing past project actuals but they don't want to know. They just tell you to reduce your estimate. Do you give in? No, you say something like this: "This is what I think it is going to cost. If you can find another project manager who will do it to your estimate please do. But if you want me to manage it this is the estimate."

Would you be taking a risk by saying that to your Board of Directors? Yes, but a very small risk indeed compared with the risk of being brow-beaten into agreeing a lower figure and then failing spectacularly. Another good example of why it isn't easy being a project manager and why it should be a well paid job.

It is quite right and proper that senior managers put in a hard challenge to your numbers: "Fifteen percent on quality? Surely that should be 5% shouldn't it?" Now he probably hasn't got a clue what it should be - he's looking for your reaction. If you immediately cave in and say you'll change it to 5% your whole credibility goes

out of the window. But if you calmly explain why it's 15% and appear to have good reason the sponsor, Board or whoever you are presenting to are much more likely to accept the rest of your numbers. The random challenge tests whether you've just made up some numbers or whether you really do know what you're talking about.



So, since you are accountable for the accuracy of the estimate, you want to get it right but how do you estimate what a stage will cost?

Bottom up estimating

1. Identify the deliverables of the stage. Sounds obvious, but consciously stop and think: what must be delivered from the stage? If we are estimating Stage 1, the deliverables might be the requirements document, the UFD, a revised cost benefit case, the Stage 2 plan and agreement, a testing strategy document, a roll out plan... and probably several other things too.

2. Write a very long list of all the work tasks you will therefore need to do in Stage 1 in order to deliver those things.

3. Estimate how many hours of work each task will take, using records of how long similar tasks took on past projects. Think of any project task you like - it's been done before on a previous project. If you knew how long it took you wouldn't have to guess.

4. Allow for 'investments'. Things like planning, time recording, team meetings, supervision, coaching.

5. Adjust for skill level. If last time a task took 1 day but was done by an expert, a novice might take two, three or even four days to do a similar task. Clearly, the hours a task will take will depend upon who you give it to.

6. Develop the draft stage plan. Sketch out who might do which tasks and when. This will usually reveal extra tasks that will have to be done but which were not on the list.

7. Assess the risks. We know roughly who will do what and when. Now is the time to take a detailed look at the stage risks. This may result in adding risk reduction tasks into the plan (remember the handover tasks for those who might leave?) plus an amount of effort/money for anything else that might

crop up that might mean tasks will have to be added to the plan once the stage is underway - in other words contingency.

8. Validate the estimate. Never use just one person to do the estimate or just one technique. Use many people and come at the estimate in as many ways as you can: top down, bottom up, third party quotes, comparison to past projects, rules of thumb, etc.

Have you got a telephone directory handy? Try this. Estimate and write down how many names you think are listed in it - just a top down guesstimate.

Now estimate and write down these 3 numbers:

- how many pages there are in the book (no peeking!)
- how many columns of names there are on each page
- how many names are listed in each column

Now multiply those 3 numbers together to give you something akin to a bottom up estimate.

Next, look at the cover of the phone book (don't open the book). It says what area the book covers. Estimate how many houses there are in that area then subtract the percentage you think may be ex-directory. If the phone book includes businesses estimate how many of them there are in the area and add them.

You should now have 3 different estimates for the number of names in the book. Now, taking those three estimates into consideration, form a revised judgement of how many names there are in the book and write it down (and remember your next pay rise depends upon the accuracy of this estimate!).

Now open the book and multiply the actual number of pages by column by names per column. (You may need to do separate calculations for the residential and business sections of the directory).

Your final estimate should be a lot closer to the right answer than you initial top down guesstimate. Your bottom up estimate (pages x columns x names per column) will probably have been the closest of your initial three estimates.

This little exercise doesn't always work, but it does show there are usually several ways of arriving at an estimate for something. Use many methods and many people, consider the results in the round and arrive at a more informed estimate.

Let us go back to that encounter by the coffee machine when you were being pressed to guess what a project might cost. Say: "I don't know, I'll get back to you." If it is a small piece of work gather a couple of colleagues round your desk, go through something similar to the phone book exercise, zip through your estimating checklist (if you haven't got one, start compiling one!) then arrive at an estimate. That number will probably be very different from the one you would have given beside the coffee machine. Of course, if you have been asked to estimate a large project it could take days or even weeks of work to arrive at anything approaching an accurate estimate: using rules of thumb, consulting records of past projects,

consulting experts, talking to potential team members, getting external quotes, etc. The time it takes to produce a reliable estimate is clearly related to the size and complexity of the project.

Mandatory review

Many companies mandate that having produced an estimate using whatever techniques are appropriate, the project manager must get the estimate peer reviewed *before* quoting the estimate to the sponsor/client/sales team/senior management.

The project manager talks through their estimate with someone from Project Support. (Note, talks through. This is not a box-ticking exercise in which you send a document to someone who checks you have filled in a particular form.) The project manager describes how they arrived at the estimate - who was involved, which past projects the estimate has been derived from or compared with, etc. Project support will ask questions such as: "Where's the change budget? Why only 5% contingency? Where's the user training time? How much time's in there for quality assurance? Why no travel costs?" Etc.

Frankly it's easy for a project manager to con the sponsor (what does the sponsor know about what should be included in an estimate?) but you cannot pull the wool over the project support guy's eyes: he's been there seen it and done it (see the chapter on project support). No longer can project managers get away with a guess, they must show someone independent that they have arrived at the estimate professionally. (You're no doubt thinking that you would *of course* arrive at your estimates thoroughly and professionally without such a check. No doubt you would. Checks like this aren't for you, they are for those colleagues of yours who might otherwise be tempted just to guess...)

The project support person will also rapidly tune in to the errors people commonly tend to make when estimating and can issues guidance, e.g. don't forget to include time for team meetings which is usually between x% and y% of project man hours.

It takes maybe half an hour to go through an estimate with project support - more for a very large project obviously - but if mandatory estimate reviews are put in place it can result in an overnight and dramatic improvement in the accuracy of estimates.

Inherited estimates

Have you ever inherited an estimate and had to manage a project to it? Was it a pleasant experience? The only person who should be allowed to quote an estimate is the person who will manage the project. That is the only person with an incentive to get the estimate right.

If you do take over a project and an attached estimate, in the first week review everything and if the estimate is wrong shout loudly (and by implication blame your predecessor). If you leave it for more than a week or so it's your fault, you caused it. Unfair, but that will be the perception.

Best advice of all is: never take over a (large) project half way through - you never know what you're taking on, you never know what stones have been left unturned that conceal nasty surprises. Go on holiday, retire, get pregnant - whatever it takes. But if there's no escape and you're forced to take a project over, in that first week lift up all those stones, even get an independent review (see the project support chapter). Maybe you'll be lucky, everything's fine. But why did the previous project manager leave?

Why people get estimates wrong

There are many reasons people get estimates wrong:

- Over optimism - nothing will go wrong
- Underestimating things like
    - control
    - coaching
    - supervision
    - project meetings
    - training time
    - communications time
    - user effort
- Lack of understanding of project scope
- Ignoring reality - assuming nothing will change, nothing unexpected will bite you
- Just guessing
- Being able to get away with no independent validation
- Estimating isn't a *managed* activity

Suggestions for getting it right

Break a big programme into sub-projects, releases, stages. Estimate each bit separately and add it all back up.

Keep the first release to essential functions. This helps in two ways. Release 1 is obviously smaller than the whole thing would have been and therefore inherently easier to estimate, but you will tend to put the stuff into Release 1 that's best understood and therefore easier to estimate. Leave all the vague stuff till a later release by which time one hopes it will have become clearer.

Don't quote narrow ranges. If someone told you a project was going to cost between 100 and 120 which number would you hear? Probably 100. But the estimator is thinking it will cost 120. Immediate mis-communication.

Quote a very wide range to indicate your level of uncertainty. "I've been looking for you, I've got to go to a Board meeting - can you tell me roughly what a project to do... would cost?" "Yes I can. It will definitely cost you between ten thousand pounds and ten million pounds." "Can't you be more precise than that?" "No, not on what you've told me so far." (One project manager carried a pair of dice for such occasions. He would roll the dice and if they showed, say, nine he'd say: "The project will cost nine." "Nine what?" "I don't know, you decide.")

Don't factor in savings from new tools. First use will probably increase costs due to the learning curve.

Have a fixed budget. This can greatly empower the project manager to say 'no' to non-essential functions. "I'd like to do that for you but I'm afraid it won't fit into the budget." If you've ever run a project you will know that some things are going to be easy to do, other things you just know are going to cause difficulties. Any excuse not to include things like that if they are non-essential is useful.

Test those who come to you with an estimate: "What is the chance the work will cost *less* than this estimate?" If the person now starts laughing helplessly what does that tell you? It's going to cost a lot more than their estimate. Ask them to try again and come back with a revised figure and tell them there must be a 50% chance the work will cost less than the estimate.

Check the estimator lives in the real world. Some people live in a fantasy world in which nobody makes mistakes, nobody changes their mind, nobody has holidays or goes sick... They are estimating what the project *ought* to cost. That figure is of no use to man nor beast. We want to know what the project's going to cost in the real world. Indeed, some experienced project managers never ask anybody for 'an estimate', they never use that word. They ask the question like this: "Please would you go away and work out for me how much you think this piece of work really will actually end up costing us?" And they say if you ask the question that way you'll get a much more realistic figure than if you asked for 'an estimate'.

Avoid automated estimating tools. The estimating tool asks you several hundred questions. Some questions you don't understand, so you guess at their meaning and give an answer. The tool gives you a very precise but very wrong answer - garbage in, garbage out!

When top down estimating, and you don't know who will be in the team, you have to assume an average skill level. But remember you will probably have more junior people than senior people so the average skill level isn't half way down, it's lower than that. If in doubt assume any A.N.Others will have the skill of a raw trainee.

Sketch out the plan/schedule before quoting the estimate. If you list all the tasks you think you'll have to do, size each one and add it up you'll get an estimate. But when you start to build the plan you will find you have to add extra tasks in to make it all fit together properly. They weren't on your list.

When bottom up estimating get team members to agree to the estimates for their tasks.

Put your estimate in writing, stating the assumptions upon which it is based. If the estimator of our wall-painting project had said: "Well, assuming the wall's prepared and there's a pot of paint and a roller there and I've got to give it one coat it'll take two hours", it would have been obvious we had quite different understandings of the project scope.

Be aware of some individual's inclinations. Some will give macho estimates: "Me? I could do that in a day easy." Others are very pessimistic: "Ooh, that could take

weeks..." Perhaps ask them how long it would take someone else to do the work: "I could do it in a day but him? About a week?"

The real secret of good estimating is recording on today's projects how many hours of work each and every project task actually takes so next time we have hard evidence and don't have to guess. More on this in the next chapter.

If you estimate properly and allow for everything we have suggested does that mean all your estimates will be too high and all projects will start coming in way under estimate? No. If you include in the estimate everything that will in fact cost you time and money estimates will simply be more accurate. If you do find that all projects come in under estimate *then* start to worry that people are padding estimates. But the risk is usually the other way: people tend to underestimate. In theory half of the projects should come in slightly over estimate and half slightly under so that across the organisation it balances out.

In conclusion:

- Good project definition is key
- Document assumptions
- Use history
- Don't guess
- Manage the process that produces the estimate
- Have mandatory independent validation
- Record actuals on today's projects to enable better estimates in future

Chapter 7 - Planning

Planning who will do what when - or scheduling as some purists would insist - is fairly straightforward if you have a team of two people. But with a team of a hundred people it can be difficult to work out who will do what and to get it all to interlock neatly. But get the plan right and your projects should run like clockwork and before long your colleagues will be saying you're the one who always gets the easy projects to manage.

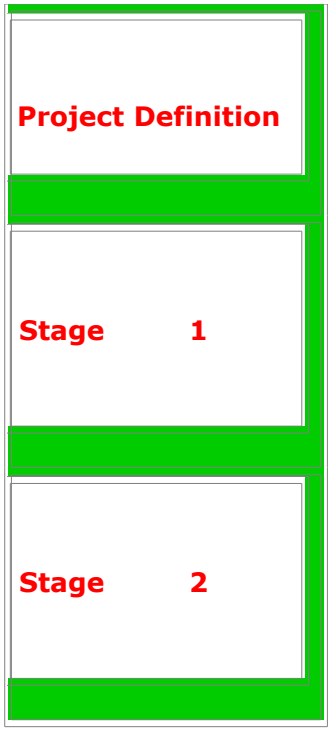
There is a big difference between planning small projects and planning large ones. For a small project a back-of-an-envelope plan may suffice but this won't work for large projects (unless you have a *very* large envelope).

Every member of a project team has a right to expect it will be clear to them which tasks they will perform, when each task should be done and what each task should deliver. In a two person project the project manager can easily plan the work for the two team members. But in a hundred person project there is no way the project manager can personally produce the works plans for every team member. On these larger projects we will need to employ team leaders to produce the detailed work plans for their teams.

In a large project the project manager will not be interested in the thousands of individual tasks in the plan: we need to find some way of summarising the project plan, and progress against it, for the project manager. And for the project sponsor an even more summarised plan is needed, maybe just showing the resource profile and key milestone dates. But what the sponsor sees should be no more than a summary of the mass of detail that will be required at the working level in a large project.

Good plans:

- help make best use of resources - a bad plan can mean people sitting around with nothing to do because something they need hasn't been produced yet.
- show each team member what they must do and when
- include everybody who has some work to do in the project - there is still a tendency among IT people only to show IT people in the project plan. The plan must include everyone: in principle if someone only has one 1 hour task in the whole project they should be shown in the project plan.
- are visible and intelligible. It is very reassuring when you walk into a project team's working area and up on the wall is a whiteboard across the top of which is a calendar of week-ending dates and down the side the team members' names and in each week some indication of what each person will be doing in that week. Why is this reassuring? If the plan is that visible and that public everyone will have critiqued it and pointed out the errors and it will have been changed so that it is now a viable and realistic plan. By contrast you get very nervous when you see a plan that you can't make head nor tail of and very nervous indeed if no-one will show you the project plan: what are they hiding? Have they even got a plan? The plan must leave nobody in doubt about what they must do and when.
- are agreed by everyone mentioned within them. Some team leaders get everyone to sign the plan to evidence the fact they've seen it and think they have a fighting chance of achieving their part of it.
- lead to better quality deliverables - put the other way around, will a badly planned, chaotic project deliver good quality? It's less likely.
- take time and expertise to produce - planning a large project is quite an art.
- are revised as the project progresses.
- are a key factor in project success. Good plan, successful project: pay rise and promotion for the project manager. A bad plan can lead to a great deal of stress during the project.



Stage planning and task size

Imagine the project on the right is a year long. At the project definition stage is it possible to plan the whole year out in detail, task by task? No. How far into the future is it possible to plan in detail? Perhaps a few months. This is one reason for breaking the project into stages. So, in the project definition stage we only produce a detailed plan for Stage 1 and a high level overview plan for Stage 2. Which implies that one of the tasks in Stage 1 will be a task to plan Stage 2 in detail. One of the deliverables from any project stage is the detailed plan for the next stage (assuming there is one).

When building the detailed plan for a stage, how large or how small should each task be? If you could choose, what would be the ideal task size? One hour? One week? Six weeks long?

If each task was one hour long there would be a vast number of tasks and you'd never keep track of them all. And what would be the problem if the tasks were all six weeks long? You'd never really know where you were. Tasks would be started and open for weeks and it would be difficult to gauge how far through they were. Overall progress would be hard to assess.

Ideally each task would be about a week long, about 30 hours work. In practice some tasks will be bigger than this and there will be some very small ones. A good sniff test of a stage plan is to see how many hours it includes, divide by the number of tasks to get the average task size. An average around 25 hours suggests the plan has about the right level of detail. An average much smaller and there are probably too many very small tasks. An average much bigger than that and there probably isn't enough detail in the plan.

Avoid long thin tasks. A task may be only 10 hours but if it's spread over 10 weeks it can cause control problems. Ideally no task will span more than two or three calendar weeks.

Each task should have an unequivocal end point: the meeting has been attended; the test case has been written; the object has been successfully tested.

Each task must be assigned to one person. If four people will attend a meeting there will be 4 tasks in the plan - one each.

The above guidance, however, only applies to what we might call 'real work' tasks. Plans, certainly for larger projects anyway, will contain two other sorts of tasks. Tasks that are fixed in time such as a team meeting every Monday morning 9am - 10am and control tasks that span the whole duration of a stage. For example, a team leader may have no 'real work' tasks to do, he just controls his team. His control task will be as long as the stage. In a small project it's all real work, but in very large projects there could be multiple layers of team leaders and sub team leaders who just control others.

So there you are at the beginning of a stage and on your desk is a very large but blank piece of paper - except at the top where it says 'The Plan'. How do you go about transforming this blank piece of paper into the plan for the stage? Where do you start?

1. Understand what must be delivered from the stage. It is surprising how often people do not think this through and thus miss deliverables. If we're planning Stage 1 the deliverables might be the requirements document, UFD, Stage 2 plan, etc.

2. On another piece of paper, write a very long list showing every single work task that will need to be done during the stage. Referring to past projects and standards will help. Team input could be very valuable.

3. Estimate, in hours, how much work each task will take. Bear in mind the number of hours will depend upon who does the task.

4. Put week ending dates across the top of 'The Plan' and your first cut of who will be in the team down the left hand side then shade out the time people will not be available to work on the project: Christmas, public holidays, holidays, courses, etc. (If someone says they're not sure when they're having holidays don't fall into the trap of putting no holiday in their plan! Guess when they might take it.) Allow for their non-project commitments and commitments to other projects.

5. In another colour as it were, shade out time for 'investments' such as team leading, coaching, orientation and meetings. Allow time for handling change requests and contingency (see below).

6. Having allowed for everything that will prevent each team member from doing 'real work' tasks, we can now turn to that list of 'real work' tasks and start the tricky business of trying to fit them into the remaining white space. This could take a lot of time and have you tearing your hair out, but stick at it. You will very likely have to add or subtract people and/or adjust the end date you were aiming for. Involve the team in this planning activity.

7. The stage plan is beginning to take shape. You now know when you will need people who are owned by other managers. You may have vague resource promises from Directors, but now you can be quite specific with people's direct managers about when you will actually need their people. We will consider how best to formalise resource agreement in the chapter on stage agreements.

8. When you believe the plan is finished, get the team together and present the plan to them line by line, task by task. Will the team find errors in your perfect plan? They will. A bit embarrassing, but better to be embarrassed one Friday afternoon than to spend the next six months having a slow motion nervous breakdown.

9. In some companies there would now be a rule that for a large project the project manager must spend an hour or so running through the plan with project support. Project support do what they can to make sure the plan seems to have been put together properly. Project support can't second guess

whether the plan is complete and correct and achievable, but if someone presents a plan to you it is usually pretty obvious whether it's at the right level of detail, clearly assigns tasks to people, includes contingency, etc. and seems to have been properly thought through. (As we shall see in a later chapter, the project support person is a peer project manager who has been there seen it and done it and they'll know whether the planning has been done properly or not.)

10. Record your plan in a tool such as Prima Vera or Track-It or Microsoft Project. We have not mentioned tools earlier for a good reason. So often team leaders get sent on an MSP course, come back and are then asked to plan a project. They open up an MSP plan and, brain totally empty, expect the plan to appear by magic on the screen in front of them. There are two totally unrelated skills at play here: one skill is knowing how to plan a particular type of project and the other completely unrelated skill is knowing how to use MSP. You could be the world's best novelist yet completely unable to use a wordprocessor. Equally you could be a Microsoft Word expert but unable to string an intelligible sentence together. Some project managers would even forbid a novice team leader from using any planning tool on their first project: make them plan on a large piece of paper and track that way too. Then they find out what planning is really all about and on their next project understand how a tool like Track-It can actually help them. Whatever you decide, avoid novice planners getting lost in the tool and losing sight of the real job: building a viable plan.

Tools

The best planning tool of all? Book a conference room. Lay the tables in a line almost corner to corner. Get a roll of plain wallpaper and lay it out along the tables. Draw lines across the roll every foot (30cm) or so to delineate weeks and write in week-ending dates. Get the team in the room, write each task on a Post-It note and have fun planning the project stage by jigging the Post-Its around until it all fits together. Roll the plan up, put it on the office wall and track progress against it using different coloured Post-Its.

We have nothing against MSP or other tools, only the tendency for them to get in the way rather than aid the planning process. Indeed, a good tool is a great asset for larger projects as we shall see in this chapter and also in the tracking, controlling and reporting chapter.

Part Timers

Have you ever had someone on your project team who was assigned for, say, 50% of their time to the project? They can be difficult to manage - you never really know when they're going to be there. Instead of nominally 50% all the way through try for something like this: you have them full time in week 1, nothing in week 2, full time in week 3, etc. Or even full time on Monday, nothing on Tuesday, etc. It is definitely worth the effort if you can get this sort of arrangement.

Critical path

If a task on the critical path slips, by definition the end date slips. Try and avoid one person in the team having all the critical path tasks in their plan and all the pressure therefore heaped upon them.

Having said that, relatively few tasks in software development projects will really be on the critical path. If you're building a block of flats the bricklayer can't lay the bricks for floor 2 unless floor 1 is there. Things have to be done in sequence. But in a software project you can write the programs in any order you like: the writing of program 2 isn't dependent upon program 1 having been written. Milestones like handover to system test are about as close as you get to a critical path task - if that slips the end date may well slip. Tools such as MSP lay great store by their critical path analysis capabilities but these are much less relevant to planners of software projects than planners of construction projects or planners of large IT hardware projects.

How long does it take to plan a project stage?

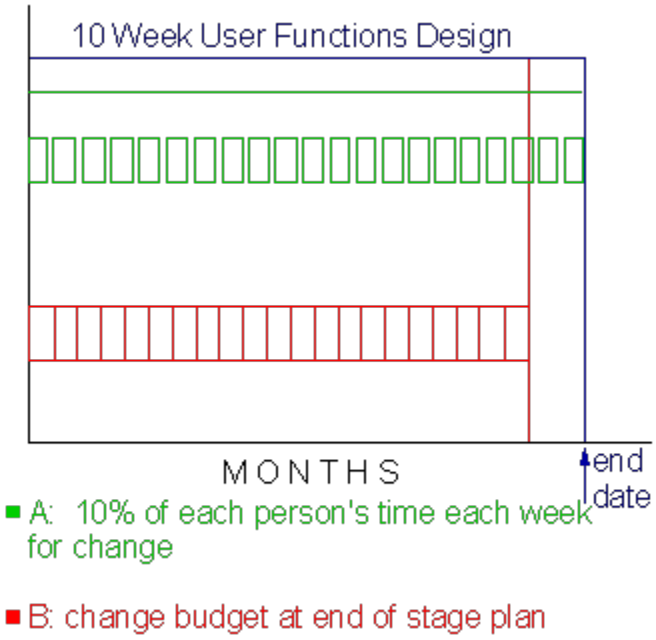
Suppose you were planning a project stage that was 5 weeks long and employed 2 people. With the aid of that envelope it might take you an hour or two to get the plan done.

But the next stage you are asked to plan is 5 months long and will employ 30 people who are owned by 10 different managers. Can that be planned in a couple of hours? No chance! This could take more like 6 weeks. Say it's mid March and the stage will begin on June 1st. You approach one of those 10 managers and ask provisionally for one of his staff full time for June. And he says yes. You ask manager 2 for a body for July and he says yes too! This is going to be easy. You ask manager 3 for a person for August but he says that person is on holiday in August, but by chance has nothing to do in June - could you take him off his hands? So you go back to manager 1 and see if you can switch his guy from June to August...

You can see that by the time you have been round ten managers - who usually won't give you quick answers - and round again, and round again - six weeks can very quickly zip by. This is not 6 weeks full time work, it is spreading the planning out over that period. But if you start early enough, by the time you get to June 1st you should have firm resource commitments in place and people should then turn up when you need them. You will then find that when you finish the project everyone will say how lucky you were - everyone turned up by magic when you needed them. It's an old adage, but: the more you plan the luckier you get. You really do make your own luck in projects.

There is only one snag with this. Suppose a critical business need pops up from nowhere. A project involving 30 people owned by ten managers must start on Monday morning - you've got 3 days to plan the project, not 6 weeks. What would you do (other than panic)? You might ask the sponsor to tell those 10 managers to come to an all day meeting tomorrow and you might ask the sponsor to attend and bring his baseball bat with him to help you extract the resource commitments from those 10 managers. This will require some quick outline planning by you before the meeting and a lot of tidying up afterwards - not to mention removal of blood from carpet. Planning a large project can be done in a short time, if painfully, but if you have 6 weeks take 6 weeks.

Change and contingency



We mentioned above the need to put change and contingency into the plan. But where in the plan should they be put?

Imagine we are planning a 10 week long user functions design step. Let's say we have decided that 10% of all the hours will be spent investigating and implementing change requests. (Change requests will be fully considered in a later chapter but to give an example: half way through the UFD step the users want to alter their (signed off) requirements, so they raise a change request.) Those 10% of the hours must appear somewhere in the plan, but where? We will consider two extreme options.

Option A would be to spread the hours out evenly, i.e. assign to each team member 10% of each week - which is half a day - for dealing with change requests. Their plan would therefore have 4.5 days work each week with, say, Friday afternoon set aside in case a change request comes along. The last task in each team member's plan is scheduled hard against the committed end date of the UFD stage.

The other extreme option would be to put those 10% change hours at the end of the UFD plan. 10% of 10 weeks is 1 week, so each team member has contiguous tasks for 9 weeks with the last week showing as reserved for handling change requests.

Which extreme do you think would be better? Before you decide, there is one important condition. Even if all the change time is at the end of the UFD plan this does not mean that all change requests are left to pile up until the tenth week. If a change request arrives in week 1 it would be given to a team member to deal with straight away, and of course all his planned tasks would shuffle along a bit to make way for the 'deal with change request' task. With option A, where half a day is left for change each week, perhaps only two or three tasks have to move along to make way for the change task.

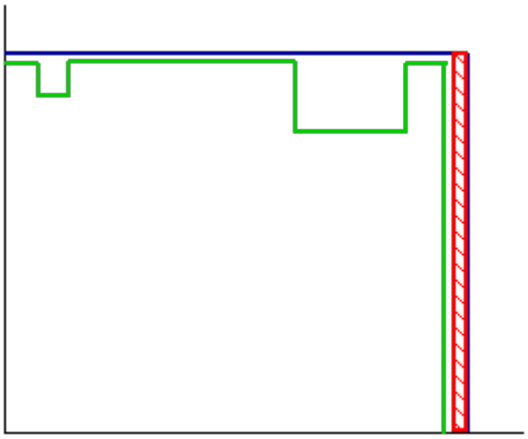
There are good arguments for both extremes but if forced to choose one or the other we would go for option B - all the change time shown in week 10. Why? Option A has half day gaps all along the plan. If no change request comes to fill a team member's half day gap at the end of week 1, will he start his week 2 task early and get half a day ahead of schedule? Everything is possible, but the risk is he won't. People usually work to whatever they perceive to be each task's deadline. Those carefully planned half days for change will just evaporate away, and when the change requests *do* arrive guess what? We haven't got the time to handle them. With option B that won't happen: everything else being equal, that last week that's reserved for change is going to be there until you choose to use it.

Luckily in practice you do not have to choose one of the extremes. What might trigger the users to raise requests to change their requirements during the user functions design step? When the users see prototypes of screen layouts and dialogue flows what happens? They will say things like: "Er, which screen is the discount code entered on?" And the designers say: "Discount code? What discount code?" And the users say: "Oops, we left that out of the requirements." Result: one change request.

Strange though it may seem one can often predict when change requests may be raised, often coinciding with things such as prototyping, reviews and sign offs. Why, therefore, might there be a mini change surge in the first week or two of the UFD step? The users are focusing their attention on the requirements document and, one hopes, signing it off. Strictly speaking it's debatable whether a change to the requirements document prior to sign off is actually a change request or error correction - either way work to change the requirements document is likely.

So, plan the change time when you think change requests might arrive and if in doubt put the change time at the end of the stage plan, with maybe a little bit of it spread all along for those few change requests that will dribble in.

In addition to the 10% for change, the sponsor has agreed that a further 5% of the effort/budget can be set aside for anything unexpected that crops up - in other words contingency. Whereabouts in the stage plan should the contingency be located? At the end. So the last 3 days or so of the UFD plan simply says 'contingency'. The UFD stage plan now has two end dates: the one the project manager has committed to the sponsor and one three days earlier which will be achieved if nothing crops up to use the contingency. Should you show both end dates to the project team? YES. Put the plan up on the wall and make it clear to the team that while you have committed the later date to the sponsor their target and their commitment to you (upon which their appraisals depend...) is the date 3 days earlier. Only if and when something crops up that you didn't expect will you, the project manager, raid contingency, create new tasks in the plan, reset the team's targets



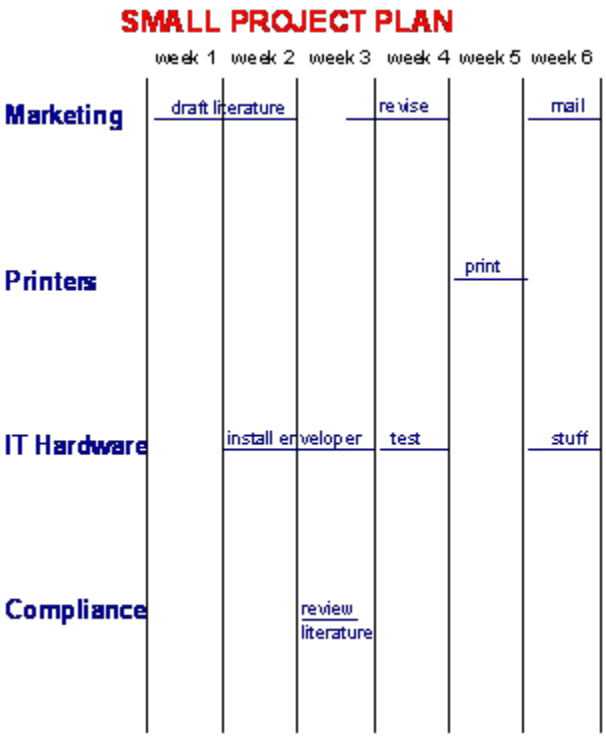
and move their earlier date a bit closer to the date you committed to the sponsor. In this way you can be honest with the team about the contingency but avoid Parkinson's Law: work expands to fill the time available for its completion.

Have a look at the picture on the right. If you see a plan with the change and contingency effort distributed a bit like this it is quite reassuring - the planner has thought about it. They have put the change effort in the plan when they think change is most likely to arrive and they've put contingency at the end. The picture represents a stage, the area above and just to the right of the green line is change effort, the red slice is the contingency. We might assume the green notch near the start coincides with sign off of the previous stage's output and the larger green notch

three quarters of the way through coincides with design reviews and prototyping. A small amount of change time has been allowed all the way along, and some of the change time is at the end of the plan, just before the contingency slice.

Very small and quite large

Project disaster lies in wait for the project manager who is "very experienced", but very experienced in running small projects. He learns over and over again a very unfortunate lesson: you don't need to bother too much about the planning. Just tell the two guys what you want them to do and that's it, that's the planning taken care of. That project manager is then given a large project to manage: he *knows* (whatever the theory says) that in reality you don't need to bother with formal planning so he doesn't and... disaster. Large projects and small projects are totally different animals when it comes to planning.



Small projects

For small projects something like the plan on the right could be all that is required. A simple bar chart that reminds each group what has to be done and when.

Or, for a team who have done many similar projects before, the plan could simply be a list of dates to remind them of the schedule this time:

- June 20: Project defined
- July 2: Market research complete
- July 16: Product spec finished
- July 20: Literature drafted
- July 30: To printers
- Aug 15: Proofs
- Sept 1: Printing complete
- Sept 15: Mailing completed

Large projects

However, for large projects something quite different will be needed. What follows in an example of a plan for a project with a team of about 20 people. Have a look at the table below - it may look like a jumble of numbers but we will go through it line by line and explain it.

Please do follow this plan through as it lays the ground work for a couple of later topics including the all important project tracking, controlling and reporting.

The table is the plan the project leader put together for himself. The first row is simply a calendar of week-ending dates - so week ending June 9th, June 16th etc.

The next row tells us how much holiday the project leader is planning to take - the numbers being hours. As you can see, he is on holiday for the last two weeks of August and for a day (the bank holiday) in the week ending September 1st. This team had a standard working week of 37 hours.

The next row shows some training courses he has planned. How do you feel about the Sickness row? In fact the planner planned every full time team member 1 hour per week for sickness. DON'T PLAN SICKNESS LIKE THAT! In our view it is quite wrong to plan sickness in this way. Sickness should be put at the end of the stage plan along with contingency but shown separately from contingency so that actual sickness time can be tracked specifically against the sickness 'plan'.

Planned Hours For: PROJECT LEADER

Task Description	June				July				August				September				October			
	09	16	23	30	07	14	21	28	04	11	18	25	01	08	15	22	29	06	13	20
Vacation	-	-	-	-	-	-	-	-	-	-	37	37	7	-	-	15	-	-	37	-
Courses	-	-	-	-	-	-	-	37	-	14	-	-	-	-	-	22	-	-	-	-
Sickness	1	1	1	1	1	1	1	-	1	1	-	-	1	1	1	-	1	1	-	1
Non-this-project time	2	2	2	2	2	2	2	-	2	2	-	-	2	2	2	-	2	2	-	2
Project Admin/meetings	2	2	2	2	2	2	2	-	2	2	-	-	2	2	2	-	2	2	-	2
Project Control	32	32	32	32	32	32	32	-	32	18	-	-	25	32	32	-	32	32	-	32
Week Total Non-project	3	3	3	3	3	3	3	37	3	17	37	37	10	3	3	37	3	3	37	3
Project	34	34	34	34	34	34	34	-	34	20	-	-	27	34	34	-	34	34	-	34
TOTALS	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37

As an aside, let us lay to rest a quite common novice's planning error with a little quiz. Imagine you work in a company that pays you for 37 hours a week. You have a one year long project. A person is assigned to your project full time for the whole year. On average, how many hours project work will that person do per week? For argument's sake, let's say a full time team member will on average do 29 project hours per week over a year.

You are now planning that person for next week. Next week he will be in the office for 5 days working full time on your project. How many hours project work should you plan him to do next week?

We hope you didn't say 29! Why is 29 definitely the wrong answer? Because it is an annual average - averaging out some weeks when he is there and some weeks when he's away on holiday or on a course or sick. So, bearing in mind that lunch time is not included in the 37 hours and time to discuss football or gaze out of the window is included in task estimates (yes - talking about football is a project activity!), in a week when he is going to be in the office should he have 37 hours project work to do?

In our example the planner would have said 35. He would argue that even a full time project team member will typically spend 2 hours a week doing legitimate non-project activities such as:

- talking to their line manager about their career (nothing to do with the project)
- reading emails that are not project related
- attending meetings that are not project meetings

The amount of non-project time will depend upon your environment and may vary by person. For example, a team member from HR, although nominally full time on the project, may have to attend an HR meeting every month, keep abreast of HR matters by reading emails, etc, whereas a contractor will have no such distractions and all 37 hours will be planned for project work (and indeed contractors may be employed for 40 hours a week so have 40 project hours planned per week.) If a team member is only assigned to the project for, say, 50% of their time they would have around 18 hours a week showing on the 'Non-this-project time' row.

The next row, Project Admin/meetings, tells us the project leader will spend 2 hours a week attending the (Monday morning) team meeting and doing admin tasks such as time recording. And the next row, Project Control, tells us he will spend most of his time controlling his team - he has no 'real work' tasks to do.

The tool has then added up for us the non-project and the project hours each week. The TOTALS row will always come to 37, except for contractors where it could be 40 and for part-time employees where it will show how many hours they are employed per week.

This second page is for a sub team leader. There were half a dozen trainees in the team and he was asked to look after them. How do we know he is a full time member of the project team? He only has 2 hours a week on the 'Non-this-project time' row. Again he has 2 hours a week for the weekly team meeting and for time recording admin, but most of his time is down to mother-henning his team of trainees. And what might PCR Investigation be? It is in fact Project Change Request Investigation time. Please look at July.

Planned Hours For: TEAM LEADER

Task Description	June				July				August				September				October			
	09	16	23	30	07	14	21	28	04	11	18	25	01	08	15	22	29	06	13	20
Vacation	-	-	-	7	7	-	-	-	-	-	-	37	37	-	-	14	-	-	-	-
Sickness	1	1	1	1	1	1	1	1	1	1	1	-	-	1	1	1	1	1	1	1

Non-this-project time	2	2	2	2	2	2	2	2	2	2	2	-	-	2	2	1	2	2	2	2
Project Admin/meetings	2	2	2	2	2	2	2	2	2	2	2	-	-	2	2	1	2	2	2	2
Team Leading	30	30	30	23	15	22	22	22	25	5	5	-	-	15	15	9	20	20	20	30
PCR Investigation	2	2	2	2	10	10	10	10	7	2	2	-	-	2	2	1	2	2	2	2
3A code duplex interface	-	-	-	-	-	-	-	-	-	25	25	-	-	-	-	-	-	-	-	-
3A code web query	-	-	-	-	-	-	-	-	-	-	-	-	-	15	15	-	-	-	-	-
3A test web query	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	10	10	10	10	-
Week Total Non-project	3	3	3	10	10	3	3	3	3	3	3	37	37	3	3	16	3	3	3	3
Project	34	34	34	27	27	34	34	34	34	34	34	-	-	34	34	21	34	34	34	34
TOTALS	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37

The planner knows something we don't know. He is predicting a surge of change requests in July and has given this team leader 10 hours a week during July to investigate them.

The next three lines we might call 'real work'. The first is a 50 hour task to be done in the middle two weeks of August. What does that suggest about his team of trainees during those two weeks? Most of them are on holiday, so the team leader can do some real work while they're away. And the planner has made an interesting assumption that by October those trainees will have had four months experience and won't need a full time mother hen anymore, so the team leader can do some real work in October as well as keeping an eye on his team. In smaller projects, the project leader would look a bit like this: a mix of supervision and real work. In huge projects there can be several layers of leaders who do nothing except control those below them.

Before we leave this team leader, please have a look at his vacation plan then go back to the project leader's plan: if you were project support and you were reviewing this project plan what comment would you make? Back.

Yup, the project leader and team leader are both on holiday at the same time at the end of August. Luckily there was in fact another team leader who looked after the shop while they were away. But this does illustrate the sort of thing it is quite easy to pick up if you are doing an independent review of a project plan.

The plan below is for one of the real workers. The plans for the majority of people in a large project would look like this. As you can see, his first task is 'orientation' which suggests he joins the team at the beginning of June. He's only with us until the middle of August and the planner has put 17 hours contingency at the end of his plan just in case he runs a bit late on his tasks. As you'd expect, almost all of his

time is assigned to real work tasks. Please look at task number 253 - how many hours should that task take him?

Planned Hours For: JOHN SPENCER

Task Description	June				July				August			September				October		
	09	16	23	30	07	14	21	28	04	11	18	25	01	08	15	22	29	06	13	20
Vacation	-	14	-	-	21	-	-	-	-	-										
Courses	10	-	-	-	-	-	-	-	-	-										
Sickness	1	1	1	1	1	1	1	-	1	1										
Non-this-project time	2	1	2	2	1	2	2	-	2	2										
Project Admin/meetings	2	1	2	2	1	2	2	-	2	2										
PCR Implementation	2	1	2	2	1	2	2	-	1	1										
279 OR Orientation	20	-	-	-	-	-	-	-	-	-										
252 CM OLT Tables	-	19	26	21	-	-	-	-	-	-										
253 3A code web OE	-	-	-	9	12	3	-	-	-	-										
254 Q3 review OE code	-	-	-	-	-	3	-	-	-	-										
255 R3 rework OE code	-	-	-	-	-	3	-	-	-	-										
256 3A test OE	-	-	-	-	-	21	10	-	-	-										
257 Q3 review OE test	-	-	-	-	-	-	3	-	-	-										
258 R3 rework OE test	-	-	-	-	-	-	4	-	-	-										
282 FD email enq UFD	-	-	-	-	-	-	13	7	-	-										
259 ID email enq IT des	-	-	-	-	-	-	-	25	31	-										
286 Q1 review enq UFD	-	-	-	-	-	-	-	3	-	-										
287 R1 rework enq UFD	-	-	-	-	-	-	-	2	-	-										
260 Q2 review enq IT des	-	-	-	-	-	-	-	-	-	6										
261 R2 rework enq IT des	-	-	-	-	-	-	-	-	-	8										
262 CO Contingency	-	-	-	-	-	-	-	-	-	17										
Week Total Non-project	13	16	3	3	23	3	3	-	3	3										
Project	24	21	30	34	14	34	34	37	34	34										
TOTALS	37	37	33	37	37	37	37	37	37	37										

It's 24 hours work over a 3 week period. And what is task number 254? It is a 3 hour task for John to sit down with other people to check his work for any errors. What, therefore is task 255? Rework - correcting any errors found by the quality inspection. How did the planner (the project leader) know the rework would take 3 hours? Was that a guess he made? It was not a guess.

On previous projects time had been separately planned for doing, reviewing and re-doing, and time recorded separately against each of those three tasks, so they knew for a fact that rework averaged around 10% of the time it took to do the task in the first place, so the planner assumed rework would be 10% this time, to the nearest hour. Of course, John Spencer might make no errors, zero rework, or he might make a complete hash of it, 30 hours rework. But across all team members across the whole stage it will probably work out to about what it was last time. This is a good example of using past factual data when planning future projects.

There are something like 400 different types of tasks in a software development project. It is surprising how few of them are fundamentally changed when switching to a new technology or a 'new' development methodology.

You can see that some of John Spencer's tasks are rather more than 30 hours, and some are 3 weeks long, but again this is a good example of a realistic plan which would easily pass the project support censor: John is in no doubt what he has agreed to do and by when.

It is also obvious that if you have a team of 20 people, planning like this will take a lot of brain time if it is all to fit together. That 3 hour task to review John's work involved two other people so they will each have had time in their plan to participate in the review during that week. Quite tricky to get it all to interlock. But invest the time and project execution will be a joy to behold.

Please don't be frightened by the chart on next page...

Any good project planning and tracking tool should be able to do something like this for you. The left-hand-most column of numbers is no more than an addition of all the hours in the plan. For example, top left, the team will have 2443 hours holiday over the duration of the project. They shall be sick for 448 hours. How do we know how sick they're going to be? Look across to the third column from the right hand side: there's a figure of 2.4. That's 2.4%. In other words the 448 sickness hours represent 2.4% of the team members' time. Why 2.4%? Well, that's how sick they were last year - we're just re-using past factual information.

The second block are project investments (not overheads!). The P1 row tells us the team plan to invest 877 hours in team meetings and admin tasks such as time recording. Please would you add up the P4, P5 and P6 hours? They come to 1253. This is the change budget - the planner thinks 1253 hours will be spent investigating, implementing and controlling changes during the stage. Please look across to the 6th column in from the right hand side and add up those three numbers (coloured green). They come to 8.7 which simply tells us that the 1253 hours for change are

8.7% of the project hours. And you've guessed it, it's 8.7% because that's roughly what it had been in project stages like this in the past.

Please imagine you are the manager of this project. We are now at the end of week 4 of this 24 week stage. According to the plan, how many hours holiday should the team have taken so far? 228 hours. And how many have they actually taken? 246 hours. So, over the first four weeks they've had 8% more holiday than they told you about. Now please look at the P6 row, PCR Control. The fourth number along is 2.42 - what is that telling us?

EFFORT DISTRIBUTION

	Hours				PERCENTAGES								
					--Of Category--			Of Proj / Non-Proj			-Of Grand Total-		
	Total	To Date			Total	To Date		Total	To Date		Total	To Date	
	Plan	Plan	Actl	A/P	Plan	Plan	Actl	Plan	Plan	Actl	Plan	Plan	Actl
NON PROJECT TIME													
Vacation	2443	228	246	1.08	61.3	41.2	45.7	61.1	40.1	44.6	13.4	7.8	8.6
Courses	226	90	97	1.08	5.7	16.2	18.0	5.7	15.8	17.6	1.2	3.1	3.4
Sickness	448	89	24	0.27	11.2	16.1	4.5	11.2	15.7	4.3	2.4	3.0	0.8
Non-this-project	882	161	185	1.15	21.8	26.5	31.8	22.0	28.4	33.5	4.9	5.6	6.5
NON PROJECT TOTAL	3999	568	552	0.98	100	100	100	100	100	100	21.9	19.5	19.3
PROJECT TIME "Investments"													

Proj admin/meets	P1	877	157	195	1.25	14.0	15.5	18.3	6.1	6.7	8.4	4.8	5.4	6.8
Proj control	P2	491	96	73	0.77	7.9	9.5	6.8	3.4	4.1	3.2	2.7	3.3	2.6
Team Leading	P3	1856	322	351	1.10	29.7	31.8	32.9	13.0	13.7	15.2	10.1	11.0	12.3
PCR Investigation	P4	607	122	105	0.87	9.7	12.0	9.8	4.2	5.2	4.5	3.3	4.2	3.7
PCR Implementation	P5	468	73	15	0.21	7.5	7.2	1.4	3.3	3.1	0.6	2.6	2.5	0.5
PCR Control	P6	178	43	104	2.42	2.8	4.2	9.7	1.2	1.8	4.5	1.0	1.5	3.6
Quality Assurance	QA	613	-	3	100	9.8	-	0.3	4.3	-	0.1	3.4	-	0.1
Int Qual check	Q1	75	40	26	0.65	1.2	3.9	2.4	0.5	1.7	1.1	0.4	1.4	0.9
Ext Qual check	QE	-	-	10	100	-	-	0.9	-	-	0.4	-	-	0.3
Technical Assur	TA	444	96	83	0.87	7.1	9.5	7.8	3.1	4.1	3.6	2.4	3.3	2.9
Staff Support	PS	640	64	67	1.05	10.2	6.3	6.3	4.5	2.7	2.9	3.5	2.2	2.3
Actions activity	PA	-	-	35	100	-	-	3.3	-	-	1.5	-	-	1.2
Investments Totals		6249	1013	1067	1.06	100	100	100	43.7	43.1	46.2	34.2	34.7	37.3
"Real Work"														
Task Type	FD	819	353	380	1.08	10.2	26.4	30.6	5.7	15.0	16.5	4.5	12.1	13.3
Task Type	Q1	64	9	-	-	0.8	0.7	-	0.4	0.4	-	0.3	0.3	-
Task Type	R1	82	11	17	1.55	1.0	0.8	1.4	0.6	0.5	0.7	0.4	0.4	0.6
Task Type	DD	1023	132	111	0.85	12.7	9.9	8.9	7.2	5.6	4.8	5.6	4.5	3.9
Task Type	Q2	111	10	4	0.41	1.4	0.7	0.3	0.8	0.4	0.2	0.6	0.3	0.1
Task Type	R2	122	5	7	1.40	1.5	0.4	0.6	0.9	0.2	0.3	0.7	0.2	0.2
Task Type	DU	124	-	-	-	1.5	-	-	0.9	-	-	0.7	-	-
Task Type	QI	16	-	-	-	0.2	-	-	0.1	-	-	0.1	-	-
Task Type	3	210	181	94	0.5	26.1	13.5	7.6	14.7	7.7	4.1	11.5	6.2	3.3

		A	1			2									
Task	Type	CO	406	-	-	-	5.0	-	-	2.8	-	-	2.2	-	-
Task	Type	Q3	494	185	123	0.67	6.1	13.8	9.9	3.5	7.9	5.3	2.7	6.3	4.3
Task	Type	R3	202	8	-	-	2.5	0.6	-	1.4	0.3	-	1.1	0.3	-
Task	Type	OR	265	259	272	1.06	3.3	19.3	21.9	1.9	11.0	11.8	1.4	8.9	9.5
Task	Type	CM	619	93	143	1.54	7.7	6.9	11.5	4.3	4.0	6.2	3.4	3.2	5.0
Task	Type	QA	58	33	17	0.52	0.7	2.5	1.4	0.4	1.4	0.7	0.3	1.1	0.6
Task	Type	CD	31	-	-	-	0.4	-	-	0.2	-	-	0.2	-	-
Task	Type	UT	31	-	-	-	0.4	-	-	0.2	-	-	0.2	-	-
Task	Type	Q4	3	-	-	-	-	-	-	-	-	-	-	-	-
Task	Type	R4	4	-	-	-	-	-	-	-	-	-	-	-	-
Task	Type	P3	15	15	20	1.34	0.2	1.1	1.6	0.1	0.6	0.9	0.1	0.5	0.7
Task	Type	ST	35	35	43	1.23	0.4	2.6	3.5	0.2	1.5	1.9	0.2	1.2	1.5
Task	Type	US	1420	10	10	1.00	17.7	0.7	0.8	9.9	0.4	0.4	7.8	0.3	0.3
Real Work Totals			8045	1339	1241	0.93	100	100	100	56.3	56.9	53.8	44.0	45.9	43.4
PROJECT TOTALS			14294	2352	2308	0.99				100	100	100	78.1	80.5	80.7
GRAND TOTALS			18293	2920	2860	0.98							100	100	100

Over the first four weeks the team have spent nearly two and a half times more hours on PCR Control than expected - 104 hours vs a plan of 43 hours. If you were the project manager would you ignore that information? You'd investigate. It could simply be a time recording error or it could be a start up problem that won't recur - both of those would be good news. But if you conclude you haven't allowed anything like enough time for PCR Control in your plan what would you do? You'd adjust the plan to allocate more hours to that activity over the remaining 20 weeks of the stage. With luck you're rebalancing - something somewhere else is running under estimate. If you're not so lucky everything is running over and you may have a real problem. What we are getting here is early warning: so far, activity by activity, how good have the estimates proven to be.

In small projects you don't need this data as you can track each person and each task individually. But beyond a certain team size you can't do that any more and you need something like this to distil the essence out of the detail so you can read the runes (to mix a couple of metaphors).

One last thing on this page: what will the fourth column from the right tell us when we reach the end of the stage? It will tell us, to one decimal place, what percentage of the project hours was *actually* spent on meetings, on change, on rework, etc. Might that data be valuable when planning the next similar project? Absolute gold dust. You've got a checklist of all the task types that were done and what percentage each was of the total project hours.

The Key Controls report contains some of the most valuable data a tracking tool will ever give you. Again, we are at the end of week 4 of this 24 week stage. Please look at block D, Total Project Hours.

KEY CONTROLS REPORT

		June 9	June 16	June 23	June 30	July 7	July 14	July 21	July 28	Aug 4	Aug 11
A. Non-project hours	Week Plan	57	196	145	170	264	306	170	96	187	147
	Week Actual	48	177	172	155						
	Week Actual/Plan	.84	.90	1.19	.91						
	To Date Plan	57	253	398	568	832	1138	1308	1404	1591	1738
	To Date Actual	48	225	397	552						
	To Date Actual/Plan	.84	.89	1.00	.97						
B. Project investment hours	Week Plan	71	313	332	297	268	281	317	395	296	365
	Week Actual	38	303	372	354						
	Week Actual/Plan	.54	.97	1.12	1.19						
	To Date Plan	71	384	716	1013	1281	1562	1879	2274	2570	2935
	To Date Actual	38	341	713	1067						
	To Date Actual/Plan	.54	.89	1.00	1.05						
C. Project real work hours	Week Plan	57	404	436	442	381	326	426	422	430	401
	Week Actual	65	438	383	355						
	Week Actual/Plan	1.14	1.08	.88	.80						

	To Date Plan	57	461	897	1339	1720	2046	2472	2894	3324	3725
	To Date Actual	65	503	886	1241						
	To Date Actual/Plan	1.14	1.09	.99	.93						
D. Total project hours	Week Plan	128	717	768	739	649	607	743	817	726	766
	Week Actual	103	741	755	709						
	Week Actual/Plan	.80	1.03	.98	.96						
	To Date Plan	128	845	1613	2352	3001	3608	4351	5168	5894	6660
	To Date Actual	103	844	1599	2308						
	To Date Actual/Plan	.80	1.00	.99	.98						

In week 4 the team should have worked 739 hours on the project but only recorded having worked 709, so it looks as though we lost 30 hours last week. However, over the first 4 weeks in total they should have worked a total of 2352 hours and they have in fact worked 2308 hours - very close to the planned number. So would you agree we have a very healthy project on our hands? Did you say "yes"? Please look at block F, tasks started.

E. Percent complete	To Date Plan	0.9	5.9	11.3	16.5	21.0	25.2	30.4	36.2	41.2	46.6
	To Date Actual	0.9	6.0	11.2	15.2						
F. No. of tasks started	Week Plan	6	24	29	25	22	25	26	19	10	5
	Week Actual	5	26	21	13						
	Week Actual/Plan	.83	1.08	.72	.52						
	To Date Plan	6	30	59	84	106	131	157	176	186	191
	To Date Actual	5	31	52	65						
	To Date Actual/Plan	.83	1.03	.88	.77						

Still happy? To date your team should have started 84 tasks but they have in fact only started 65 tasks, so they're a quarter behind despite putting in roughly the right number of hours. Please look at block I, task completed.

G. Tasks started, not due	-	1	-	2
H. Tasks overdue starting	-	-	3	21

I. No. of tasks completed	Week Plan	4	13	19	27	26	21	34	16	11	8
	Week Actual	3	10	10	19						
	Week Actual/Plan	.75	.77	.53	.70						
	To Date Plan	4	17	36	63	89	110	144	160	171	179
	To Date Actual	3	13	23	42						
	To Date Actual/Plan	.75	.76	.64	.67						
J. Tasks completed, not due		-	1	4	5						
K. Tasks overdue completing		1	5	17	26						
L. Hours for completed tasks	Week Plan	25	118	167	391						
	Week Actual	44	139	135	475						
	Week Actual/Plan	1.26	1.18	.81	1.21						
	To Date Plan	35	153	320	711						
	To Date Actual	44	183	318	793						
	To Date Actual/Plan	1.26	1.20	.99	1.12						
M. Hours ahead of schedule		-11	-65	23	-177						

How do you now feel about your project? Your team should have finished 63 tasks by now but have only finished 42. Ouch.

One last number: near the bottom right hand corner of the table is a figure of 1.12. What is that telling us? The 42 finished tasks took on average 12% more hours to complete than estimated. What rather worrying thought is in the back of your mind about all the remaining tasks in your plan: will they all run over by 12% as well? Maybe they will, maybe they won't, but today at the end of week four we have some investigating to do. We're getting early warning of a possible overrun here, we should not ignore it.

Please take a few moments to study this Key Controls Report to get a feel for what it is telling us. In the tracking, controlling and reporting chapter we will explore how reports such as this provide the raw material for crisp and concise project status reporting.

But where do all these nice numbers come from?

It relies on each team member recording each week how many hours they have actually worked (paid or unpaid) on each project task, and for unfinished tasks how many hours it will take to finish them.

Have a look at Frank Barlow's time sheet. How is Frank doing so far on the project?

WEEKLY TIME SHEET (including current status): Frank Barlow

Task No.	TASK DESCRIPTION	DAILY HOURS						Total Hours	Hours Left	Hours Ahead Last wk	Planned Hours This wk	Planned left Next wk
		W	M	T	W	T	F					
100	Web path 1 code inspection									-1	0	0
101	Web path 1 code rework									-3	0	0
102	Web path 2 code inspection									-1	0	0
103	Web path 2 code rework									-2	0	0
104	Web enquiry 1 coding									0	8	0
105	Web enquiry 2 coding									0	13	0
	Project Admin/ meetings										2	
	Team Leading										6	
	Inspecting others' work										4	
	Vacation										0	
	Courses										0	
	Sickness										1	
	Non-project time										3	

TOTALS								

Frank is 7 hours behind schedule and should have finished the first 4 tasks last week. The 'Planned Hours This wk' column tells us how he should be spending his 37 hours this week. Let's hope he puts in a bit of extra effort and does those tasks as well as catching up the 7 hours and 4 tasks from last week. Clearly, this time sheet shows a mixture of data from the project plan and Frank's actual status as of last Friday. (The first **W** under DAILY HOURS stands for Weekend.)

The time sheet captures hours actually worked on each task rather than chargeable or paid hours. If it took 90 hours to do a task we want to know that. Depending upon your environment you may also need to record paid hours and paid overtime hours for charging purposes.

If the project team is working unpaid extra hours but are running on schedule, should we care? We should. We are relying on the team's goodwill - which has its limits. And if they work excessive hours to keep on schedule diminishing returns set in, and eventually they may get so tired that the lid suddenly blows off the pressure cooker.

Most time recording tools allow individuals to enter their time directly. However, this can result in made up numbers being recorded and the data not being used.

An alternative is to have the team leader speak to each member of his team on a Friday afternoon for a five minutes one-to-one discussion about the team member's week and to go through their hard copy time sheet. You might even want to consider insisting that the team leader keys in the time sheets for each and every one of his team - that way you know the team leader has at least seen the data. The team leader should also quality assure the numbers: if he suspects Frank is making up random numbers, visit Frank at the end of each day to go through his time record for that day. He'll soon get fed up with that!

Time recording has to be sold to the team and there are a couple of ways of doing it. "Team member, I've estimated this task at 30 hours but frankly it is a guess. If it actually takes you 90 hours or whatever please let me know by recording that on your time sheet so next time I'll know to estimate 90 hours and it'll be less stressful for you next time around." Secondly, show the team how you will use the time sheet data to track and control progress. If the team see there's real value in accurate time recording they are more likely to do it.

We are primarily recording *project* time on these time sheets. In principle if someone worked on three projects during a week they will get visits from three team leaders on Friday. Time sheets are a project tracking mechanism: if someone is assigned to your project for, say, 20 hours a week we only want to know how they spent the hours they worked on the project, we're not interested in what else they did during the week.

Miscellany

Even if you have a plan a bit like the one we have been through, still buy a whiteboard and put a summary of the plan up on the wall so that everyone can understand at a glance what is going on. If the team is spread across several sites also buy a webcam.

Large projects can be divided into 'work-intensive' and 'milestone' projects. In a large work-intensive project there are many people mostly working full time on the project - the sort of plan we have been through is appropriate in those circumstances. However, some very large projects in terms of budget are of a different nature. In these projects a succession of high cost items is delivered and installed. The project plan consists largely of milestone dates and the project manager's main job is chasing suppliers to make sure they deliver as promised. Even if several people are involved and they are doing critical things the number of hours they will work on the project is relatively small. This sort of project needs a milestone chart or a critical path network diagram rather than a mechanism for counting and analysing hours and tasks.

When planning a long stage - say 5 or 6 months - it may not be desirable at the start of the stage to plan the whole stage at the task level. One way round this is to have 'block' tasks. For example a block task of 250 hours for a team member in months 5 and 6 that covers code, review, correct, write test data, test, and handover of a part of the system. Perhaps only in month 2 or 3 would the planner break this out into 2 coding tasks, 2 review tasks, 2 correction tasks, etc, i.e. the one block task becomes 14 work tasks. This is fine but it can look as if the plan is growing because the number of tasks is increasing. Counting the block task as 14 tasks from the outset can overcome this.

Some planners plan all the easy tasks at the beginning of the stage. Their rationale might be that this aids team morale. But all the hard tasks come at the end and they're the ones that tend to overrun. If you can, plan the chewy tasks at the start and plan the easy, predictable ones at the end of the stage.

If you publish a design document on June 1st, does the design stage end on that day? Probably not. What happens when any document is published for review and sign off? Nothing. Which means you must go and chase people to read the document. And then they start asking questions, want things explaining, request changes, etc. If you assumed your team were going to be available full time to work on the next stage you will get a nasty shock: they keep getting pulled back to do these tidy up activities. At the end of Stage 1 allow effort for tidy up that overlaps with perhaps the first two weeks of Stage 2. Stage 1 actually finishes two weeks after Stage 2 has started. More on this in the tracking, controlling and reporting chapter.

Always plan and track in hours, never days or half days. People will argue how long a day or a half day is but an hour is an hour is an hour.

Net vs gross can be a source of confusion. Firstly an hour is an hour is an hour - there are no net hours or gross hours - just hours. For the purpose of illustration let us assume that a full time project team member will do 29 hours project work a week on average over the course of a year. There are 4.333 weeks in a month, so the person will on average do 29 x 4.333 which is around 125 project hours a month. This is sometimes referred to a gross man month - a full time team member will do

125 man hours project work per (gross) man month. This is useful for working out how many people the project will need: the project requires 49 gross man months of effort? That's about 7 people on average for 7 months. Usually when people refer to man months they mean gross man months. If people do use terms such as net man days or gross man weeks ask them to explain exactly what they mean - a pound to a penny they won't be able to.

We have seen how very small projects can have simple plans and we have also seen the sort of detail that will be needed for larger projects. Only you can gauge what will be most appropriate for your project.

Whether large or small, well planned projects tend to cost less, take less time to do and deliver better quality than badly planned ones.

Chapter 8 - Stage Agreement

We are about to begin a stage. We have built the plan and we have verbal agreements from resource owning managers to supply the people and other things (hardware, office space...) that we need. Why bother to write it down and get written commitment, don't we trust their word?

People have an amazing ability to 'forget' what they promised verbally. Put it in writing in a Stage Agreement and get resource owning managers to sign it.

It is completely unreasonable to expect a project manager to commit if he has no way of securing firm commitments of the resources he will need in order to deliver.

It's easy for a senior manager who sits atop an empire of 300 people to promise the 6 people a project manager wants. It is quite another thing to get the immediate managers of those 6 people actually to release them.

Before the project manager commits he must be confident that the resources he needs will be available. Getting the IT team members committed to the project is relatively easy: they exist to do projects. It is usually much harder to get business users' time committed to the project: they've got other things to do like run the business.

A stage agreement is first and foremost an agreement between the project manager and those who must provide resources for a project stage. It's a bit like a mini PDD - a PDD for a stage but with the added ingredient that resource providers sign it.

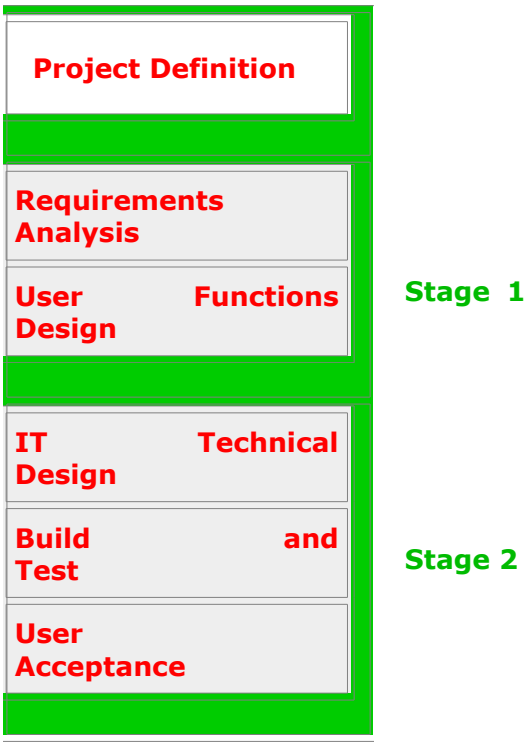
Stage agreements also help resource owning line managers. If you are a Marketing manager with 20 people reporting to you and you're providing people to 5 projects you will have signed 5 stage agreements and it will be clear to you what you've given away and therefore what's left to carry out your department's day to day work.

A stage agreement should be no more than 2 or 3 pages and should contain:

- Distribution list
  - resource owners for sign off
  - others for information
- Description of stage
- Deliverables from stage and who will sign off to accept them
- Completion criteria: how we will know that the stage has finished
- Success criteria: how we will know whether the stage was successful
- Roles and responsibilities, including junior roles that will not appear in the PDD
- Risks that threaten the success of this stage and how they will be managed
- Resources needed for this stage
- Cost of this stage
- Key milestone dates (one every 2 or 3 weeks)
- Management and reporting: how change will be controlled, progress reported, etc.
- Quality plan: how the quality of the stage deliverables will be assured
- Outlook for later stages: indication of resources that might be needed for subsequent stages

If you *know* how you'll manage risks, report progress, control change, etc summarising it in a stage agreement won't take very long. If you don't know how you'll do those things you're not ready to start the stage. The ability to sit down and write a stage agreement in an hour or so is a revealing test of readiness.

Let's write a stage agreement for Stage 1 of the project on the right.



Stage Agreement for Stage 1 of Project XYZ

Distribution

- G Smith (HR manager) for resource sign off
- J Jones (Marketing manager) for resource sign off
- B Gates (IT manager) for resource sign off
- R J Patel (IT infrastructure manager)
- Steering committee members for information
- Project role holders and team members for information

Description of stage

- This is Stage 1 of project xyz and it comprises requirements analysis and user functions design.

Deliverables and sign off

- Requirements document April 10th
    - to be signed off by G Smith, J Jones and R James (project user manager) by April 24th
- User Functions Design document May 14th
    - to be signed off by G Smith, J Jones, R James and R J Patel by May 28th
- Stage 2 plan and stage agreement May 14th
    - to be signed off by resource owners (to be identified by May 1st)
- Stage 1 completion report May 28th
    - to be signed off by H E Goodboss May 28th

Stage completion criteria

- Above sign offs achieved
- Sponsor sign off to concur that stage has completed

Stage success criteria

- All issues resolved within 14 days of documents being published and sign offs achieved as scheduled.

Roles and responsibilities

- Sponsor: I Topman (Director, Operations)
- Steering Committee: Ivan Empire (Marketing Director), J Goodheart (HR Director)
- Project Manager: H. E. Goodboss
- Project User Manager: R. James
- Requirements Team Leader: P. D. Q. Finklestein
- Design Team Leader: P Boseman
- A project hierarchy chart showing every team member can be found on the intranet at www.ourco.xyz.roles.htm

Risks

A full risk register for Stage 1 and for the whole project can be found at www.ourco.xyz.risks.htm. The main risks threatening the success of this stage are:

- Lack of user resources:
    - written resource commitments are sought via this document

    - the sponsor has agreed personally to intervene should resource owners in the event be unable to fulfil their commitments
- Lack of office space where the team can be co-located:
    - Jason Fixit (Facilities Manager) is pursuing a refit of the old canteen for our use
    - In the meantime the team will use the Executive Board Room

Resources needed for this stage

The attached week by week, person by person plan shows precisely who is needed and when. In summary, resource owners will supply the following:

- G Smith 2 people full time March 1 - May 28
- J Jones 1 person on a one week on, one week off basis starting March 1 and ending April 20 (see plan for details)
- B Gates 1 person March 1 - May 28, second person April 10 - May 28. These two will continue through to the end of Stage 2 (which is the end of the project which we hope will be October 30th) as development team leaders.
- Project Support will conduct a Health Check around the end of April. (This is included in project support's plan which has been committed to the Exec.)
- R Patel has committed to make the development environment available by June 10th. Although this date falls after the end of Stage 1, commitment is sought now due to its criticality.

Cost of stage

- $400K. This is primarily people's time costed at Finance Standard Rates. The development environment cost of $45K will be included in the Stage 2 cost (current best guess $600K). See the business case www.ourco.xyz.buscase.htm for a full cost and benefit breakdown.

Key dates

The attached plan shows milestones in red. The major milestones are:

- Stage start: March 1st
- Requirements workshops completed: March 30th.
- Requirements document published: April 10th.
- Requirements document signed off: April 24th.
- Design simulation completed: May 12th.
- Design document published: May 14th.
- Plan and agreement for Stage 2 completed: May 14th. (And Stage 2 starts on May 14th.)
- Design signed off, all Stage 1 deliverables signed off, Stage 1 end: May 28th.

Management, reporting and communications

- weekly team meetings run by team leaders
- weekly reporting by team leaders to PM (Mondays)
- fortnightly reporting by PM to sponsor

- reporting pro formas can be viewed at www.ourco.xyz.reps.htm
- issue and change management will be handled personally by the PM given the tight timescales. Procedures and forms at www.ourco.xyz.issue_pcr.htm
- risks will be reviewed in weekly meetings. The project risk procedure and risk register is at www.ourco.xyz.risks.htm

Quality management

- The requirements document will be fully inspected by relevant user personnel (see attached plan that shows inspection meeting dates and attendees).
- The UFD design document will be inspected and a 3 day simulation will be conducted.
- Our quality processes are documented at www.ourco.xyz.quality.htm and are almost a direct copy of those used by J. Clever on the recent rxt project (thank you, John).

Outlook for later stages

- Stage 2 (build, system test and UAT) will start on May 14th (a 2 week overlap with Stage 1's tidy up period) and will complete around October 30th. Contract software developers will be recruited and on board by May 1st and so have 2 weeks for education in our standards and project orientation, and they will sit in on design inspections and simulations so they know what the project is all about before they start 'work' on May 14th.
- We anticipate we will need 2 user testers from HR and 1 from Marketing for the period Sept 1st - Oct 21st, dates to be firmed up and committed in the Stage 2 agreement by May 14th.

A real stage agreement wouldn't have many more words than this one. This document is not your entry for the Pulitzer Prize, it is a summary of what is already known by and agreed to by all parties.

If you saw a stage agreement like that, signed by the resource providers, to which was attached a plan of the sort we explored in chapter 7 you would probably have this feeling: the project manager knows what he's doing.

Suppose there was a rule in your company that before any stage could start you, the project manager, had to make a 20 minute pitch to the sponsor essentially presenting the contents of the stage agreement to demonstrate that things were planned, resources committed, etc. Would you go to that meeting saying: "er, the plan isn't quite done yet, we haven't got any resource commitments yet and the costs are being looked at by Finance..."? You wouldn't. As the date for that sponsor presentation approaches it focuses the mind wonderfully and is a real incentive for the project manager to get off his backside and get things sorted out.

How would you persuade the sponsor that resource owning managers had committed resources? Show him their sign offs (in most organisations an email suffices as a sign

off). Even better than that, bring those managers to the meeting and have them commit eyeball to eyeball to the project sponsor (and then sign). Much harder to renege on *that* commitment.

And in this short meeting, how would you persuade the sponsor that the plan is viable and realistic? Present a one page summary plan signed by every team member as evidence they think they can do it. That is very persuasive. Compare it to past projects. Show evidence that project support have looked at it and OKed it. And bring a ton of backup charts showing every single work task and threaten to show them all if he still doesn't believe you!

But what if a few days before you are due to make this presentation to the sponsor you email G Smith, the HR Manager inviting him to sign the stage agreement to confirm his verbal agreement. There is no reply, so you go to see G Smith but you can't get hold of him. You find his boss who says: "resource commitment? That's news to me you need to talk to G Smith." Tomorrow morning you are due to make the presentation to the sponsor either to ask permission to start the stage or to explain that start must be delayed because you haven't got G Smith's sign off.

You ring the sponsor asking if you can delay the meeting as you haven't got G Smith's written resource commitment. A half decent sponsor, satisfied that you've done what you can and now facing a delay to the start of his project will say: "leave it with me".

G Smith gets a call from the sponsor asking whether he will be able to supply the 2 people as verbally promised. If you were G Smith would you now want to be saying: "I'm sorry, I know I promised them but I didn't really think about it and actually I can't supply them." You would not want to be in that position.

But if you knew that would be the consequence of making empty promises you'd probably think a lot harder about it before making even a verbal promise.

As project manager you want to find out *before you commit* whether resources really are going to be available or not. In the example, if those 2 HR people really cannot be made available the go no go meeting will have to be delayed while you do some speedy replanning, see what can be done with resources that *can* be made available - which may mean higher costs and later delivery dates - and then go to the sponsor with a revised, well founded plan that you have a reasonable chance of achieving.

In reality we all know people in our own organisations whose promises we know will be fulfilled. And we may know people who will promise almost anything we ask for without really thinking about it. If you suspect the latter, you would not leave it until a last minute panic, you would be talking to that person a week in advance, helping them think about their resources, other commitments, etc helping them understand whether they really can support your project, so that when they do promise you're fairly confident they will be able to deliver.

If there is a rule in place: 'no sign off no start', it does focus minds and make it less likely your plan is built on sand. When you commit it's based on rock-like

commitments made to you. Written commitments won't guarantee resources will actually turn up as committed, but it does make it a lot more likely. 'No sign off no start' is like a nuclear deterrent: you hope you never have to use it but the threat must be credible.

Either the resource is committed and the project manager commits on that basis or the resource cannot be made available and the project manager replans and commits on some other basis.

For large projects it is worth considering having the project sponsor send the stage agreement out requesting resource commitments with a covering note saying something like: "Thank you for agreeing to support this key company project. Please confirm your commitments as documented herein to me in writing within 2 days." So resource providers commit in writing to the sponsor - they will probably be less inclined to renege than had they committed to the project manager. Of course, the project manager has drafted the sponsor's note and all the sponsor has done is agree that the note can go out in his name.

Programmes

We have described how the project manager demonstrates readiness to start a stage by a short presentation to the project sponsor.

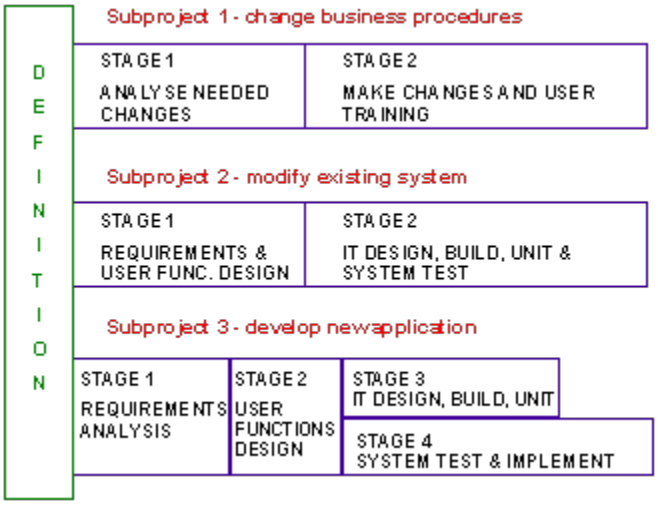
The same principle can be applied by managers of programmes: before each stage of each sub project can begin the sub project manager makes the same sort of presentation to the programme manager. If several sub projects are starting at the same time have all the sub project managers present at the same meeting. This will sometimes identify overlaps and gaps: two sub projects thought it was their job to produce the UAT plan - or worse, nobody was going to produce it.

It may well be that in practice there is no option but to start, so the presentation becomes a readiness review. But if you were the programme manager and a sub project manager came to you seeking the go for a sub project, but it is abundantly clear that the plan is half baked, estimates are vague and no firm resource commitments are in place you might:

- a) note this as a black mark for the sub project manager's next appraisal (unless there are clear extenuating circumstances)
- b) realise that by allowing the sub project manager to break the 'no sign off no start' rule you are now a little exposed
- c) give that sub project some help and put it on special watch until you are confident the deficiencies had been rectified

Benchmark, foresight, relationship to PDD

The stage agreement also provides a benchmark against which the project manager's performance can be measured: he has said very clearly what will be delivered at what cost by when and with what resources.



If you were a line manager and a project manager asked you for two people to do some testing starting tomorrow morning, you might try and do what you can to help but politely point out to the project manager that next time would he please give a bit more notice. A stage agreement mandates and formalises that notice.

In a large project or programme there will be an overall PDD which outlines the many sub projects, stages, etc. If you are managing one of those stages you have to pin things down in much more detail than the overall PDD ever will.

Time goes from left to right in the picture on the right. This project/programme has three sub projects and 8 stages in total so will end up having 8 stage agreements.

Sub project 1 has no IT involvement. In Stage 1 of sub project 1 the business will consider how they might re-engineer the manual part of their processes and Stage 2 of sub project 1 - if approved - would implement those changes (new office layout, cross-skilling, training, etc.).

Sub project 3 is a fairly large software development project. Note that in this sub project Stages 3 and 4 will run in parallel. Stage 3 (technical design, build and unit test) done by the IT team, and in parallel with that Stage 4 (preparing for and performing system test and implementation) performed by a separate team made up of user and IT people.

In a project with two stages, the PDD and the Stage 1 agreement will be produced at roughly the same time towards the end of project definition, and there will be a degree of overlap in their contents - senior role players will be listed in both. By the time the Stage 2 agreement is produced a few months down the track, even some of the senior role players may have changed.

In small projects which only have one stage, do not produce two separate documents. A combined PDD/stage agreement is all that is needed, the added ingredient when compared to a PDD alone is the sign off from resource owning managers. This is a good example of scaling the project management controls to the need of the project. Indeed, in very small projects a couple of sides of A4 may serve as the PDD/stage agreement/project plan all rolled into one. Don't smother those tiny ones with unnecessary bureaucracy.

Monitoring commitments

Suppose it's March and your stage agreement contains a resource commitment for June - three months away. Between now and June it may be a good idea to bump into the resource owning manager every now and again and say: "Are we still OK for Fred Smith in June...?"

And if the manager moves and is replaced, go see the new manager and get him to re-sign the stage agreement - otherwise when June comes he'll say: "Sorry, that's news to me..."

Clearly, there will need to be sanctions against those who commit resources in writing but still renege. If a manager signs to commit a resource then at the last moment refuses to supply the body for no good reason (up to that point having told the project manager the resource will be available as promised) it would be quite unreasonable for the pain to be inflicted upon the project manager - assuming he has done all the right things to secure the resource. In this case the sponsor should ensure that the reneging resource owner feels the pain via his next appraisal.

A stage agreement is in essence a summary of the stage plan and the minutes of the meetings you had with resource owning managers. It is a tool to be used by the project manager to make it more likely the project will succeed. Put other things in the stage agreement if that will help you. Leave suggested things out if there is no need to including them. This is a document to help you succeed, not a bit of bureaucracy to keep the methodology police happy.

Stage agreements help in several ways

- Writing things down reduces vagueness
- Secures resource commitments
- Sign offs flush out false promises
- Promotes understand of what the stage will do
- Documents agreements already reached verbally
- Increases the chance of project success

Chapter 9 - Project Support

Why do projects fail? Usually because they weren't planned properly, risks weren't managed, roles were unclear, etc. In other words inadequate project management tends to be the underlying cause rather than technical failure.

The aim of project support is to help projects start out with things adequately planned and to provide advice during project execution. In this chapter we will consider what project support do at the start of, during, and at the end of project stages. We will also look at things they may do at any time.

At the start of a stage

Unless a project is very small we would recommend a rule that the project plan must be reviewed by project support before the sponsor is asked to make the go no go decision. It takes a couple of hours to go through:

- estimates: how were they arrived at?
- schedule: does it exist, has the team signed up to it?
- risks: have they been assessed, is there a plan to manage them?
- stage agreement: are resources committed?
- change and issue procedures: are they set up?
- reporting: are adequate tracking and reporting mechanisms in place, is it clear who will communicate what to whom?
- quality: are quality assurance activities included in the schedule?

The aim is not to catch the project manager out, it is to help him ensure that the project is, in the broadest sense, properly planned and therefore more likely to succeed. This is not a box ticking exercise: have you got a document called a Risk Management Plan? Yes? Tick in the box. Next.

This is much more: talk me through how you're going to manage risks. "Yes, that sounds good and would it also help to assign each risk to someone to look after...?" This comes down to consulting an expert who can add value. It is not the methodology Thought Police checking bureaucratic compliance.

If the project manager has got the project adequately planned the review won't take long. If the project manager is inexperienced the review may add a lot of value. But the fact that everyone running any significant project *must* go through the review means fewer projects start with half-baked plans. And common sense would, one

hopes, make it occur to less experienced project managers to invite project support to advise them while they are doing the planning so there are no surprises at the mandatory plan review.

So, in the round, project support help project managers start each project stage with better plans than might otherwise have been the case. A project support review does not mean the plan is perfect and guaranteed to succeed - it only that were possible!

During a stage

We will describe two types of independent reviews of projects: health checks and informal reviews.

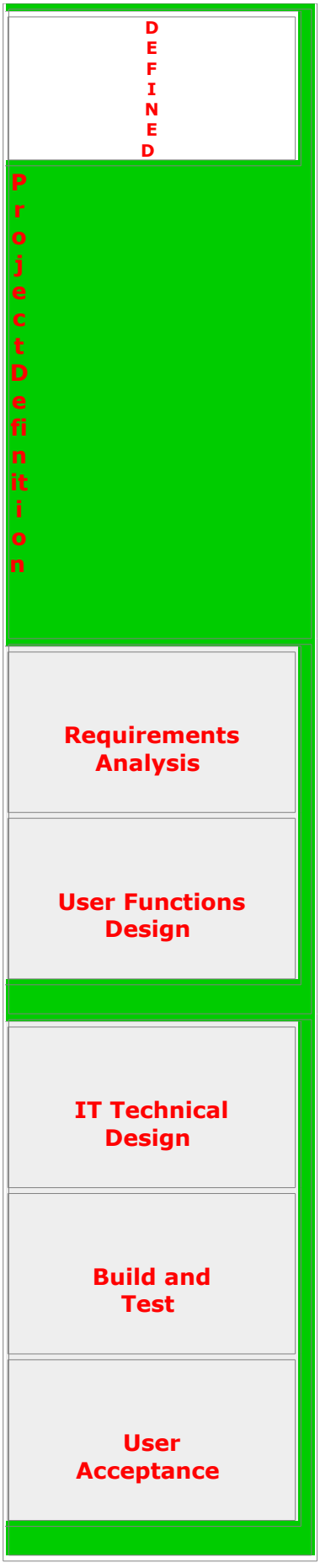
Health checks - which projects and when

The Board or project sponsors decide which are their most significant projects and designate these as key projects. These will tend to be the large, business-critical or high risk projects.

The rule then is that any project designated as a key project must plan in at least one health check. For a software project, half way through the UFD step is a good time to do a health check: there's enough track record from which to draw conclusions and, one hopes, enough time to rectify any shortcomings.

Health Check----->

Health Check----->



Long projects may have a second health check planned in, perhaps half way through the build stage.

Note that the health check is *planned in* - this is not a surprise dawn raid by the secret police. We shall see that the health check aims to offer advice to the project manager and also provide the sponsor with independent verification of project status. It will also become clear that health checks only really work if project managers see them as adding value and therefore do not hide things from the health check review team.

Health checks - who does them

Our favoured approach is that the health check team consists of 3 people:

- a project support person to lead the review
- a project manager from another project
- a team leader from another project

There are (at least) two other ways of building review teams each with pros and cons:

1) 3 people from project support and/or internal audit or similar groups conduct the review. The pros might be that they get very good at doing reviews but they are open to the criticism: "what do they know about the realities of running projects, stuck up there in their ivory tower?"

2) 3 project managers or project leaders from other projects. The pros are that they should be able to offer good, pragmatic advice given that they are also running projects in the real world, but the old pals act may mean problems don't get into the health check report and senior managers are none the wiser.

The mixed team approach ensures consistency and rigour - that the process is followed and problems are reported - and that some real world pragmatism is injected.

A week or so before the health check, the project support person who will lead the review meets with the project manager to discuss possible areas of project weakness. It is then the job of the project support person to choose review team members who can add most value in those perceived areas where improvement is most needed. So, if the change control process seems to be causing problems project support seek out someone with expertise in that area who is best placed to offer good advice.

Health checks - typical schedule

- Monday afternoon. The review team trawl through the project's written records: plan vs actual data (such as we saw in the planning chapter), change logs, risk register, etc.
- Tuesday morning. The project manager makes a full disclosure presentation of status, problems and planned actions, issues, major risks, and so on. Warts and all.
- Tuesday lunchtime. The review team go into a huddle and begin to formulate their assessment and identify further information they need.
- Tuesday afternoon. Vitally, the review team is empowered to speak to anyone they wish to: the project sponsor, the users, trainee developers, or whoever. Essentially this is fact finding.
- Wednesday morning. The review team discuss their findings and recommendations with the project manager.
- Wednesday afternoon. The review team write the health check report.

For very large projects, and those scattered across many sites, it may take longer than this, but the schedule above gives an idea of how long it might normally take. Bear in mind two things: health checks are only done on larger projects and the work entailed in being reviewed is in the project plan.

Health checks - what is checked

- The project plan. Remember, the health check review is being conducted half way through a stage. A project plan must exist - we know this because project support saw it before the stage began. But sometimes the plan looks great on paper but when you go back half way through and talk to the team members it's obvious they don't really understand what the plan is telling them to do and they're ignoring it. So the review team establish whether the plan is as good as it looked, is being followed and is being updated as the project progresses.

  Imagine this. On the Monday afternoon the written records reveal that the stage is half way through and tasks completed so far have all, on average, cost 25% more than expected.

  On Tuesday morning you ask the project manager: "what are you doing about this overrun of 25% on task costs to date?" And the project manager replies: "Oh, that's interesting, I didn't know about that...". What might you write in the report? (And keep it clean!)

  The temptation is to say: 'The project manager is not managing the project' or 'the project is out of control' or something like that. Both may be true, but never write anything like that in the report. If you say the project manager is not controlling the project he may say that he is, but in a different way. The point is: keep the findings factual and unarguable. So you might say: 'the project manager did not know that completed tasks were on average 25% over budget.' That is a fact which, one hopes, the project manager will not dispute.

  What recommendations would you make to the project manager? Perhaps:

    - every week sit down with the team leaders and review the numbers coming out of the tracking tool: there is valuable information which is currently not being exploited.
    - revise the plan to reflect the cost overrun to date.
    - go and have a chat with the project sponsor as it looks on current trends as though the project budget will be exceeded.

  A health check report might contain a dozen or so findings - each a sentence or two long - and a bulleted list of recommendations for each. So the report is not a huge document and may even be in the form of presentation charts.

- Change. Is it being controlled? And even if it is, is so much change being taken on board that project success is being jeopardised?

- Quality management. When dates start to slip what tends to get sacrificed? (No, not the project manager...) Quality. All those finely planned quality checking tasks in the plan get skipped. And quality problems go undetected until later in the project and then they cause much bigger delays.

  So the review team check that the planned quality tasks (see the quality management chapter) are being properly performed despite pressure on dates.

- Roles. Are role players stepping up to their responsibilities?

- Resources. Are promised resources being supplied?

And you may be thinking of a dozen other things you would want to look at if you were conducting the health check review.

One thing health check reviews *do not* do is to examine work products (such as the requirements document) to try and assess their correctness and completeness: the health check review team neither have the time nor the expertise to do that. What they *do* examine is whether those who were planned to check these products for correctness have actually done so.

Health checks - report

The health check report contains these things:

- summary and rating
- findings (e.g. finished tasks have cost 25% more than expected)
- recommendations (e.g. review figures weekly with team leader, speak to sponsor)
- rules and guidelines recommendations
- health check process (who participated)
- project manager's comments (how useful the review was to the project manager)

When the health check team review a project they may conclude that the project is in great shape but isn't following some of the project management rules because it is

different from most of the company's projects. The review team don't recommend that the project should be shoe-horned into the rules but that the rules be modify better to help such projects in the future. This provides effective feedback from the project management coal face to project support who are the custodians of the rules and guidelines.

The health check team form a judgement about how likely they think it is the project will meet its commitments. One way of expressing their opinion is the A, B, C, or D rating.

- A: we think the project has every chance of meeting commitments, potential problems are under control.
- B: the project will probably meet commitments so long as actions are put in place to address identified problems.
- C: major problems exist which will cause project failure unless significant action is taken urgently.
- D: the project will not meet its commitments.

Health checks - follow up

The follow up will clearly depend upon the rating.

To ensure findings are not simply ignored, following a B, C or D rating the project manager must in subsequent monthly reports to the sponsor report each finding until it has been actioned.

However, after a C or D more urgent action will be needed. Bearing in mind health checks are only done on the larger, more business critical projects, a C or D rating could be an exposure for the company, let alone the project. So the follow up after a C or D might be as follows.

Within 3 working days of the health check the project manager must meet face to face with the sponsor and explain the problems and the actions to be taken. And no doubt the project manager will get help from the sponsor and other relevant senior managers.

Strange though it may seem project managers sometimes positively welcome C or D ratings. Why might this be? Here is one example: A project manager had a resource commitment. The date arrived but the resource didn't. The project manager went to the resource owning manager who told him to go away. The project manager escalated to the resource owning manager's manager who also told him to go away. The project manager escalated to his own line manager for help in resolving the problem, but his boss said: "can't you deal with that?" Nobody wanted to know. The health check happened then to come along and gave the project a C rating. The sponsor received an independent report saying he was the only person with the authority to fix the problem. The sponsor ignored the report. The project failed. An audit was then carried out and the sponsor was asked what he did when he received the report and he replied "er, nothing..." and an audit finding went against the sponsor's name.

This is a rather extreme example but it does make the point that an independent report should be enough to galvanise senior management into action when they might otherwise be tempted to ignore the problem and hope it will go away. In our example, in a perfect world, the project manager would have gone to the sponsor with a clear statement of the problem seeking his support before the review team turned up. Though in a perfect world all projects would succeed and health checks wouldn't be needed.

Project managers may also welcome C or D ratings when they have just taken over a project. The project's in a mess, the project manager asks for a health check a) to get experts to help him identify the problems and b) to make it clear to the world that what he's taking over really is in bad shape.

And project managers may even ask for health checks even if they have managed the project from the start: the project manager is courageous enough to realise he is out of his depth and needs help, so asks the experts to come and do a review and help him identify what needs to be done.

Following a C or D, replanning is usually required and new budget and date commitments will probably need to be agreed with the sponsor. A follow up health check will be scheduled 6 weeks or so later to assess whether the project will at least meet its revised commitments. One hopes this second review will be an A or B - revised commitments will be met. Getting occasional Cs or Ds happens - all part of the fun of being a project manager. But getting two Ds in a row can be a little painful for the project manager.

Health checks - anecdote

One afternoon the project support manager was bumped into at the coffee machine by a developer. After the usual pleasantries the developer asked: "is a review planned of our project?"

What was the developer telling the project support manager? You should come and do a review of our project...

The project support manager spoke to his boss and said that although no health check was planned for that project he thought it would be a good idea to do one. His boss said: "that's what I pay you for, go do it". The project support manager approached the project manager and asked him if he would like a health check done on the project. The project manager was, to put it mildly, hostile. What would you have concluded if you had been the project support manager?

After some discussion with their mutual boss a health check was conducted. It was abundantly clear the project manager had been suppressing the bad news, not addressing problems and misreporting status. The team were all getting thoroughly disenchanted. The boil was lanced, corrective action was put in place. Suppression of difficult reality shouldn't happen but it does. Occasionally project support can in this way be a useful safety valve, releasing the pressure before projects spectacularly explode causing collateral damage to all in the vicinity.

Health checks - audience

Following an A or B rating a one line email goes to relevant senior managers: nothing you need worry about. After a C or D they get the full report or better still a presentation from the health check team. Senior managers should then ensure the project manager takes appropriate action and they will be aware of what they need to do to help get the project back on the straight and narrow.

Having said all of that, the health check report is first and foremost for the project manager, offering constructive and practical advice on how they might improve the project's health.

Informal reviews

Health checks, as described above, take a few days and are only appropriate for larger projects. For medium sized and less critical project an informal review can be valuable to the project manager. One person, maybe a project support person or perhaps another project manager, spends half a day with the project manager at his desk looking at the sort of things a health check would look at and offers advice, guidance, suggestions, etc. No written report, no A, B, C, D rating. Look upon this as a mentoring process.

Informal reviews can sometimes pose a dilemma though. If you did an informal review of a colleague's project and it was obvious to you the project was going to fail but the PM has been hiding the truth what would you do? You'd probably strongly recommend to the PM that he tells the boss and point out that if he doesn't you have a duty to. Otherwise you become an accessory.

Health checks - musts for successful reviews

To return to formal A, B, C, D health checks. The 5 Cs for successful health checks:

- Consensus. The review team strive to reach agreement with the project manager on findings and actions *before* they write the report.
- Constructive. The review must be seen to offer useful, constructive advice on how shortcomings may be addressed. Try also to comment on what the project is doing well.
- Courteous. Never write in the report: "the project manager is an idiot". Dispassionately report problems and actions.
- Confidential. Don't go around the building telling everyone what a mess the project is. Keep ratings, findings and conclusions to those who have a right to know (sponsor, steering committee, etc).
- Concise. Keep the report succinct

Health check - benefits

Do project managers ever lie about project status? Of course not. But occasionally they do forget to mention certain things and inadvertently leave the wrong impression in the sponsor's mind. However, if the project manager knows a health

check is coming in 2 months time he is more likely to report the problems himself today: if he hides them and the review flushes them out the sponsor is going to ask: "why didn't you tell me...?" Health checks should reassure the sponsor that the project really is as healthy as the project manager says it is.

In the couple of weeks before a health check the project manager will often get around to dealing with things he has been putting off. Indeed, it may well be that half of the corrective actions get taken before the review team even shows up.

Even if you are your company's best and most experienced project manager you will learn something by reviewing another project. You'll spot some really elegant technique they are using. And what will you do? Steal it and use it on your project.

And sometimes another member of the review team will highlight a problem with the project being reviewed and you will realise you have exactly the same problem and you'll nod sagely and then scurry back to fix your own project.

Inviting a novice team leader to sit in as a 4th member of a review team provides a great learning experience. He is forced to see in great detail how all these project management things are done in reality.

Health checks are a good way of spreading best practice and good ideas across an organisation.

Bottom line, though: health checks save money. Sometimes a health check brings to light a problem that would otherwise have lain dormant. The sooner you spot them the cheaper it tends to be to fix them.

Health check - project managers' reactions

The project manager should be obliged to write a comment in the report about how useful the review was to him. One hopes they might write things like this:

- encouraging to see real problems identified
- health check helped me focus on problems in estimating and controlling
- very sensible recommendations, I would like to express my appreciation

But that isn't quite always how they react. Occasionally there is a dispute. The review team says it's C or D the project manager says the project's fine and should be A or B. Who would you put your money on? Back the review team. Sometimes the project manager can't see the wood for the trees and sometimes they are in psychological denial - they won't even admit to themselves it's a mess.

Review teams are not infallible, they don't always spot all of the problems. Project support need to develop a checklist of things to look at, questions to ask, etc. Where a project fails despite recently scoring an A or B rating, project support do need to take a look to establish why the project went wrong, whether warning signs could in fact have been picked up at the health check and if so to update their processes so they don't miss similar things in future health checks.

Compliance monitoring

This is something quite different from health checks or informal reviews. Compliance monitoring involves spending half an hour with the project manager making sure he is adhering to the company's project management rules:

- Have you got a signed off PDD? Yes? Tick.
- Have you got a signed authorisation to be spending this stage's budget? Yes? Tick.
- Have you got a stage agreement signed by resource providers? Yes? Tick.
- Have you got a risk register? Yes? Tick.

This is largely a waste of time so project support should only do these compliance checks very occasionally if they believe too much backsliding is creeping in. This type of compliance check tells you next to nothing about the project's health.

Project support's attitude may be summed up thus:

- If you're breaking all the rules and going to succeed that's interesting and we'd like to know how you're doing it.
- If you're following all the rules and failing that's a shame and we'll come and help you.
- If you're breaking all the rules and going to fail you're in deep doo-doo.

Safest bet: follow the rules and succeed.

Technical Reviews

The health checks described above do not, by and large, look at work products to gauge their correctness (they look at whether the team is doing the things they planned to do to ensure their work products are correct).

Technical reviews do look at a project's work products. For example, in a leading edge IT project, a technical review team may spend a week or more examining in detail the system design to assess whether it will work, will give the desired performance, etc. Whereas health checks are conducted by people with project management skills, the technical review team will be technical experts. However, a technical review finding that says: "The XLQ to TDQ link object will not allow transference of TIF image data" will have non-technical senior managers' eyes glazing over. They will have no idea whether individual technical review findings are showstoppers or minor suggestions. So, for technical reviews, the review team rate *each finding* A, B, C or D as well as giving an overall rating.

Other things project support should do

- Collect, collate and disseminate lessons learned (this will be covered in the stage and project end chapter).
- Write, own, update, publicise, explain and help projects intelligently adhere to the company's project management rules and guidelines.
- Organise project management training.
- Run a project management forum. A quarterly open house meeting at which project support, project managers and team leaders share experience. For

example: "we're finding in health checks that this... is a very common problem, this is how you can avoid similar problems..."
- Maintain an intranet forum for project management hints, tips, rules of thumb, etc.
- Write an annual report to senior management summarising how projects went over the past year. For example: "Over the past 12 months 63 projects had this problem... Board of Directors, you need to take this action... otherwise it'll happen 63 times next year..." Tell senior managers what needs to be done to make the company a more project-friendly environment.
- Be a free consultancy resource. Arguably this is the most important role of all. See the next paragraph.

Who fills the project support role

Who are these paragons who inhabit project support?

The most junior team leader must feel able to consult project support and ask the dumb questions they may not want to ask their boss. They won't do this if the project support person is too senior.

People in project support must be seen to have been there, seen it and done it: they must have been involved in projects and so have that credibility.

They do not need to be experts in everything, but they do need to know who the experts are so they can point enquiries in the right direction. "You want to know the best way of reporting system testing stats in a games development project? Have a chat with Lara Homestead, she's the expert on that."

Project support must be seen to be 'us', part of our family. So, if 'we' are an IT shop, project support must reside in IT and be staffed by IT people. If 'we' are marketing, project support is in marketing and staffed by marketing people.

In some companies project support begins life within IT, and they own the rules that govern how projects are run. But when senior business managers begin to appreciate that projects are not IT things but are the way the company makes it future, they decide that project support should move and report to someone other than the IT Director and be staffed by people with relevant project skills whether they be from IT, marketing or wherever.

Project support should not be part of an Internal Audit function. The "when the auditors come and see you answer honestly using only the words yes or no but don't volunteer any information" mentality will not result in project health checks that add value. Internal Audit might perhaps audit the processes used by project support to satisfy themselves that the project community has in place appropriate self-policing procedures.

The best people to have in the project support role are team leaders who have been personally involved in major disaster projects: they know the problems, they know the warning signs (which were probably ignored on their project) and they know the

pain. And they should be naturally helpful people who want to help their fellow human beings avoid similar pain.

As a rule of thumb there should be one person in project support for every hundred people involved in projects. In a large company with two thousand people in projects perhaps project support will be 15 strong comprising a couple of heavyweights to lead health checks of major projects through to several team leaders who are assigned to project support for eighteen months or so. In a small company with only 20 people doing projects one of the PMs may be nominated to spend, say, one day a week doing some of these project support things but obviously much less formally.

Project support do not assist in the day to day running of projects. Project administration and reporting is the job of the project manager and his project team. And do not populate project support with methodology theologians.

Reviewing project plans and conducting health checks requires judgement, the ability to offer useful advice and the ability to formulate and present controversial conclusions. This can be difficult. Compliance monitoring - checking projects have the right bits of paper in place - is easy. But plan reviews and health checks add value.

When the first health checks are done in a company it is possible that they are almost all Ds. Not because all projects will fail but because project planning and tracking is so poor the review team conclude they have no idea whether the project will succeed or not and they conclude the project manager hasn't either. A better way to improve project governance is to embark upon an education programme to teach project managers and team leaders how to manage projects (don't send them on courses that teach them how to pass a methodology exam), write a starter set of in-house project management rules and guidelines (which takes days not months), put in place start-of-stage plan reviews, and then start doing formal health checks.

Project support must be empowered by senior management - Directors, sponsors, etc - to conduct reviews on their behalf.

The Appendix contains a sample project support job description.

In conclusion

Project support is there to:

- Help project managers
- Reassure sponsors of large projects
- Help identify problems early - ideally during project planning
- Make it more likely that more projects will succeed

Chapter 10 - Tracking, Controlling and Reporting

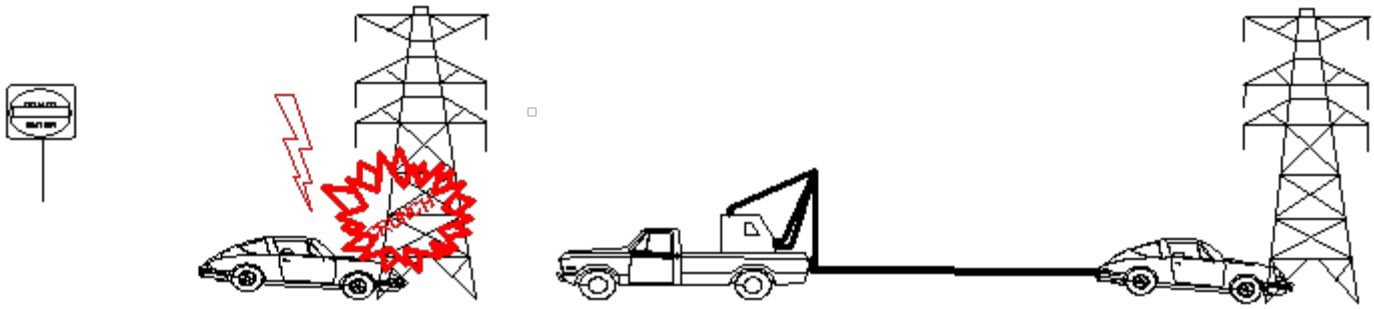
Project control is very simple in principle. It boils down to knowing where you are and what you have done, comparing that to where you should be and what you should have done, spotting the difference and deciding what to do about the difference. Simple as that. Shall we move on to the next chapter?

Make your friends groan with this one. It works better spoken than written. Controlling a project is great fun if you remember the three basic laws of project management. Do you know what Murphy's Law is? "If it can go wrong it will". Do you know what Sod's Law is? "It will go wrong in the worst possible way". Do you know what Cole's Law is? "Coleslaw is finely chopped cabbage in mayonnaise."

The old adage that no news is good news does not apply to projects. An information black hole conceals one thing only: bad news. And if you have data about the project you could interpret in several ways only the most painful interpretation will be the correct one. This is no place for optimists.

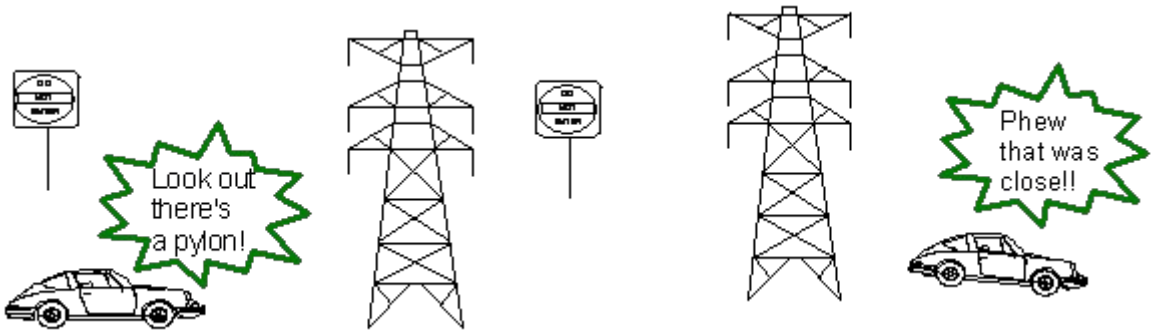
What is the purpose of project control? It helps the project deliver as committed: dates, costs, quality, etc. It also enables accurate reporting of project status. Clearly, we must be able to spot deviations from the plan and sort them out, but the real art of project control is looking forward to foresee potential future problems so that we can avoid them. This Porsche driver has had a bit of a plan deviation, you haven't got to be a genius to figure that out and even the corrective action is fairly self evident.

Identifying a plan deviation **Taking corrective action**



But we want to be more like this next Porsche driver who has a sensitive, forward looking problem alert mechanism (a pair of eyes). He sees the pylon coming, makes a small course correction and just skims past it. This is what good project control comes down to - spotting the problems in advance and avoiding them rather than discovering them the hard way.

Identifying potential problem **And avoiding it**

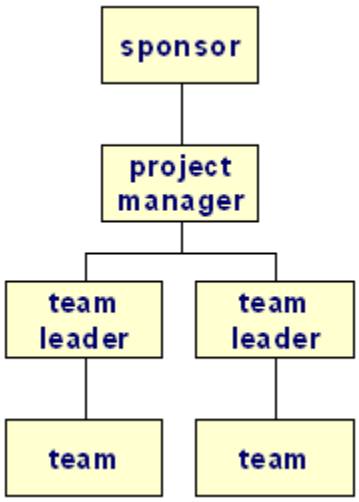


Control boils down to 'are we on plan?' which implies we have a plan. But imagine we have a plan that only contains half the work we will actually need to do. Knowing we are bang on plan is highly misleading. For good control we need a good plan.

Where are our commitments written down? In the PDD, stage agreement and the plan itself. But suppose we are half way through a 5 month long stage. Are we still tracking against the plan we built two and a half months ago? Almost certainly not - over those first two and a half months we will have had to change the plan and we're now tracking against that changed plan.

Who authorises changes to the plan? Maybe you're thinking 'the sponsor'. Imagine at the start of a stage we told the sponsor it would cost 10,000 hours of work. But at the end of week 1 we realise we have made a mistake: we don't think it will cost 10,000 hours, we think it will 10,001 hours. Should we get the sponsor to approve the 1 extra hour? No. But suppose at the end of week 1 we don't think the stage will cost 10,000 hours we think it will cost 20,000 hours, should the sponsor approve the extra 10,000 hours? You bet.

Where should the boundary line be drawn? There should be a predetermined tolerance limit - only variances outside that limit need be approved by the sponsor. If the tolerance is too small the project manager would spend all his time getting tiny variances approved. If the tolerance is too wide the sponsor has little or no control over costs. Before a stage starts the project manager agrees how much he can change things by without seeking sponsor approval - i.e. his tolerance limits. The project manager must then decide how much of his tolerance he will delegate to those below him in the chain of command. The sponsor may have the freedom to move the end date by a month without seeking the approval of anyone more senior, but would you allow that same tolerance to the most junior member of the team? Probably not. The amount of tolerance you have depends upon who you are. We will discuss some actual amounts a little later.



Bear in mind also that tolerance is not on the current position vs the plan but on where we are going to end up. So, if we're currently 25% over budget but nevertheless confident of completing the stage on budget, no approval is needed. If we are currently on budget but think we will end up 25% over budget that requires approval (unless we have a very generous tolerance limit indeed!).

Let us consider the tracking, controlling and reporting activities that we would expect team leaders, project managers and project sponsors to perform.

Team Leader

What should a team leader do to keep a grip on the work for which he is responsible?

- Monitor team's status. Know exactly where the team is versus their plan.
- Check tasks really are completed. Particularly at the start, and for new team members, if someone says a task is finished check that it really is finished until you know you can trust what they say.
- Responsible for quality of team's work. Individuals are accountable for the quality of their work but if someone in your team delivers bad quality that will feature in your appraisal as team leader too (see the quality management chapter).
- Keeper of project records. On large projects the project office will look after most of the record keeping but on smaller projects it often falls to the team leader(s) to keep the records straight.
- Collect weekly time sheets. Once a week establish precisely where each team member is versus their individual plan.
- Keep the team work plan up to date. This is a thorny one. At the extremes team leaders come in two flavours. At one extreme are the theoretical perfectionists. At the end of week 1 of a 4 month stage they spend the whole weekend moving tasks planned for months 3 and 4 around for some theoretical precision in the plan. And at the end of week 2 what do they do? Move them back to where they were in the first place and shuffle several hundred other tasks around in pursuit of that theoretical perfection. The other extreme, however, is much worse. The lazy sods. Once the work has started they never update the plan so after the first few weeks the plan bears no

resemblance to what is actually happening and is never seen again. Somewhere between these two extremes must be a happy medium. Each Friday the team leader captures actual status from each member of their team - tasks finished, hours worked, etc. The team leader then revises the team plan so that at least over the next 6 weeks or so the plan reflects exactly what we *now* think will actually happen over those next 6 weeks. Critical path tasks more than 6 weeks out should be replanned if we *know* they're going to be late, but as long as the plan is roughly right between 6 weeks out and the end of the stage leave that part of the plan alone. Do whatever is right for you, but don't waste time being a theoretical perfectionist and don't be lazy.

- Weekly team meeting. The team leader gets their team together for an hour once a week. Each person says where they are and what they are doing. Risks, issues and changes relevant to the team are reviewed.
- Report status weekly to the project manager. See below.
- Liaise with other team leaders. Keep your team's work synchronised with other teams'. Escalate to the project manager if you can't.
- Agree tolerance limits with team members. The team leader may have an understanding with each of their team members rather as follows: As long as you can complete your tasks by their scheduled end dates I will be happy. If that means you must work a few more or a few less hours than planned just get on with it. But if you will not be able to meet any task's deadline let me know and I will change the plan.

The team leader manages the work of his team.

Team leader's tolerance

At the start of a stage your team leader says to you: "boss, I've heard about this thing called tolerance which I understand to mean that I can change my plan in some ways without getting your prior approval - but I'm not sure what I can and can't change..." How would you as project manager respond, what would you let your team leader change without your approval? Maybe you're thinking: 'nothing at all'. But if he's told you the stage will cost 10,000 hours and wants to change his plan so that it shows 9,999 hours you would doubtless be happy for him to do that. And a change to 10,001? Perhaps you'd set something like this as his tolerance limits:

Stage milestones and end date:	0% tolerance
Stage cost:	5% tolerance
Hiring and firing:	0% tolerance - he can't hire and fire team members
Change of project scope:	0% tolerance - he can't unilaterally reduce scope in order to meet dates etc.
Quality:	0% tolerance - he can't sacrifice quality to meet the date (now there's a novelty!)
Dependencies:	0% tolerance - he can't change his plan such that it would screw up other parties

How much freedom has the team leader now got to change the plan without your approval? You may be thinking 'not a lot', but the answer actually is a huge amount.

The team leader can switch tasks between team members, resequence tasks, make some tasks bigger than they were others smaller. He can make a lot of change to the plan without getting anywhere near any of those tolerance limits. There are two points here: first you are explaining to your team leader what he is allowed to do but much more important you have just explained why you pay his salary: he *must* move things around if that would improve the outcome. He can't change key dates, unilaterally descope, hire and fire, screw up dependencies, sacrifice quality or go more than 5% over his budget but within those limits he must manage to best effect.

If you do not have this five minute discussion with novice team leaders they are sometimes frightened to change anything in their plan in case they breach a limit they didn't even know was there. You are empowering them - indeed obliging them - to change their plan if that would be beneficial.

(Quiz. Six team leaders each have 5% cost tolerance on their bit of the plan. They all use their 5%. By how much has the overall budget increased? 5%.)

Project manager

What should the project manager do to keep a grip?

- Keep his finger on the pulse - wander about, talk to people informally.
- Weekly meeting with the team leader(s). See below.
- Manage issues and changes - this is covered in the next chapter.
- Monitor stage agreement commitments - as we mentioned in the stage agreement chapter, don't let managers forget what they have committed to you.
- Consolidate sub project reports. If there are just 2 team leaders this is not too difficult. If there are 4 sub projects each with 6 team leaders the project manager will need a project office to add it all up. This does not mean the project office reports status to you (or to anyone else), it means they add up some numbers for you. Better still, in a project with, say, 8 team leaders, designate one of them as the senior team leader, or perhaps project leader, and have him consolidate the plan and status for you. Keep ownership of the plan and status reporting strictly in the chain of command, don't hive it off to an admin support office. If you do that you may find the team leaders start to disown and even disdain the plan and status data.
- Revise the plan. If one team can't make their dates you may need to move resources between teams or direct another team to move their dates to reflect the first team's slippage.
- Control contingency. As we discussed in the planning chapter, the last 5 days of your stage plan may initially be labelled contingency. When there is no alternative, you give some of that contingency elapsed time to a team leader, they reset their plan and their team's targets and let's say you've then got 3 days contingency remaining. The trick is to sell to the project team the tightest possible task by task plan you can get them to sign up to at the beginning of the stage and to hold in your back pocket things like contingency. Lash them regularly like galley slaves into meeting those

aggressive targets, and only when you conclude the targets are genuinely impossible do you give them some of your contingency and reset their targets. Moving dates in this way is not slippage or failure, it's how you avoid Parkinson's Law. If you and the team meet the date you committed to the sponsor you have succeeded. If the team think they can spend the contingency whenever they want to they'll spend it for you ten times over in the first fortnight!

- Report monthly to the sponsor. See below.

These days some sophisticated clients of large software development projects *demand* that their suppliers include contingency in the project plan and budget and show them (the client) where it is in the schedule. Then every month report how much of the contingency has been used and for what, and therefore how much contingency is still left. This level of maturity is rare among sponsors in process-based companies - we all have an informal education job to do to explain that contingency is there to help *them* make *their* project successful.

Project office

The project office has no part to play in controlling the project.

All they do is administer the processes, add up numbers and maybe physically produce the report you'll show the sponsor. This is not to say their role is unimportant. On the contrary it's vital. On a large project if change requests aren't administered properly there could be chaos - but the *management* of change, i.e. making decisions about change requests, is the project manager's job. Similarly, the team leaders produce, revise and own the plans for their teams - their plan is a tool they use to help them manage the work of their team - it is not a bit of administration to be shuffled off onto the project office.

The project office no more runs the project than the Managing Director's secretary runs the company.

Project sponsor

How does the sponsor know what's going on and keep overall control of the project?

- Monthly status meeting with the project manager (or more frequently if he so chooses)
- Health check reports
- Stage go no go authorisation with agreed tolerance e.g.
    - +/- 1 week on the end date
    - +/- 10% on stage costs
- Authorising revised budgets and dates

Let's say you as project manager have committed a stage cost of 10,000 hours to the sponsor and you have 10% tolerance - you *must* come in between 9,000 and 11,000 at the end of the stage.

But you realise at the end of week 1 that it won't be 10,000 it will cost 12,000. You've only spent tuppence so far, but you must inform the sponsor now of this projected overrun.

The sponsor reluctantly agrees to approve a revised budget of 12,000 hours with new limits of 11,000 and 13,000. But at the end of week 2 guess what? Sorry, sponsor, I think it's going to cost more like 15,000. The sponsor will conclude you don't know what you're doing.

Going back once within a stage with your begging bowl happens all the time. Going back twice as above will be a bit painful for the project manager. But the pain of going back twice is nothing compared with the pain of coming in after the end and telling the sponsor you've overspent the upper tolerance limit. If you do that you get hung, drawn and quartered (and then fired). This is an encouragement to come forward during the stage as soon as it becomes clear you'll need more budget. The sponsor must be able to control what is spent in his name before it is spent.

Bear in mind the budget, 10,000 in our example, *includes* the change and contingency budgets both of which are estimates of the amounts of money you think you are actually going to spend, in exactly the same way that the travel or testing budgets are estimates of amounts of money you think you are going to spend.

If the project will under run its 9,000 lower tolerance limit, why should the project manager have to come forward and tell the sponsor? So he can spend the money elsewhere. (Please email us if you ever see this happen.)

Weekly status meeting

Each Monday morning the project manager meets with the team leader(s). Keep the meeting informal and relaxed but at the same time factual and objective. So it's not: "it's fine boss, don't worry...", each team leader reports:

- tasks done or not done versus the plan
- hours worked versus plan
- problems and how they are being dealt with by the team
- status of issues and risks
- how the team leader has re-jigged his plan. Even though the re-jigging has been within their tolerance they keep the project manager - and other team leaders - informed about how they have changed their plan.
- problems the team leader needs to escalate to the project manager
- permission to change things which are outside their tolerance limits, e.g. to move a milestone.

It might take half an hour or so per team leader, so it'll be a long morning if you have 6 team leaders.

The project manager should tell the team leaders about the discussions he has had with the project sponsor and get the team leaders to pass that to the team members. This is often overlooked, but it can aid team morale and motivation if they are kept in touch with what the bigwigs are talking about.

One benefit of tasks that are about a week long is that they are either done or they are not done - end of debate. With long tasks one can spend all Monday morning debating how far through they are this week. Long tasks tend to remain 90% complete for about half their lifespan. Never have anyone report 'percent completed' for a project task, it's meaningless.

Minute these weekly meetings. Just the facts reported by the team leaders, permissions given and actions agreed.

Clearly, if major problems or issues arise during the week the team leader shouldn't wait for the Monday morning meeting to tell the project manager.

Whole team meeting

Every now and again get the whole project team together. How often depends on many factors. In a large project perhaps a monthly whole-team meeting (and more frequent meetings of sub teams as discussed above). Sometimes, perhaps at critical times in the project, a ten minute project team meeting at the start of each day may be appropriate.

Monthly status meeting

Once a month present status to the sponsor. We give an illustration of this below.

In a large project there will be an extra layer of management. There will be a project director, below him project managers and below them project leaders and or team leaders. In this case the project director may hold a monthly meeting at which each project manager formally presents the status of their (sub) project. Some project directors like to run this monthly meeting with all project manager and all team leaders present and the reporting actually isn't done by the project managers but by one of their (well rehearsed) team leaders on their behalf. Sometimes called leapfrog reporting. Why do it this way? Well, it's excellent experience for the team leaders and helps them develop their project management skills. But some project directors feel they get a truer picture of what's going on from a team leader than they would from a project manager. Project managers sometimes put a gloss on things whereas team leaders tend to tell it like it is. This sort of forum also aids cross-project (cross-programme) team building and communication.

Project Monthly Report (PMR)

Let us see what would be shown to the sponsor (or project director, as in the previous paragraph) in the monthly status meeting.

Please imagine you are the project sponsor. Have a look at this and see how you feel about your project:

PMR for Project: Project XYZ	**Report Month: June**
Project Manager: I.Goodboss	**Author: A.T.Leader**
1.STATUS SUMMARY vs LATEST APPROVED PLAN	**RAG status: GREEN**

YES	NO	
yes		Will the Project products meet specifications?
yes		Will products be delivered on approved schedule?
yes		Are External commitments being honoured?
yes		Are External relationships satisfactory?
yes		Will the project milestones be met?

You have a big grin on your face, you are being told everything is wonderful. But have a look at a bit more of the report:

PMR for Project: Project XYZ		Report Month: June
Project Manager: I.Goodboss		Author: A.T.Leader
1.STATUS SUMMARY vs LATEST APPROVED PLAN		RAG status: GREEN
YES	NO	
yes		Will the Project products meet specifications?
yes		Will products be delivered on approved schedule?
yes		Are External commitments being honoured?
yes		Are External relationships satisfactory?
yes		Will the project milestones be met?
+ 10%	- %	Project hours to date as +/- percentage of plan
+ %	- 33%	Completed Tasks to date, as +/- percentage of plan
+ 20%	- %	Hours for Completed Tasks as +/- percentage of plan

Would you now believe the green status and the yes column? What are we being told:

- we've spent 10% more hours to date than we should have
- we've completed a third less tasks than we should have
- completed tasks have taken 20% more hours to complete than we expected

This project cannot be green, the yes column cannot be correct.

As a principle we should never report only our opinion nor should we only report bald numbers - always both, just as in company accounts.

Project managers often track and manage work on the basis of hours worked and tasks done. However, we also need to know how much money has been spent. If we only have half the number of people we should have, we could be way underspent on our budget to date. At the same time, if we have more (expensive!) contractors than expected the costs of tasks completed could be way over in cash terms.

Whether money spent is just people's time converted to money (by the project office) or money spent is only cash expenditures such as contractors, equipment and travel, or both added together, or whether each is tracked separately, etc is one of the myriad decision you must make before your project starts.

Please have a look at section 3 Outlook To Completion:

PMR for Project: Project XYZ		Report Month: June
Project Manager: I.Goodboss		Author: A.T.Leader
1.STATUS SUMMARY vs LATEST APPROVED PLAN		RAG status: AMBER

YES	NO	
yes		Will the Project products meet specifications?
yes		Will products be delivered on approved schedule?
yes		Are External commitments being honoured?
yes		Are External relationships satisfactory?
yes		Will the project milestones be met?
+ 10%	- %	Project hours to date as +/- percentage of plan
+ %	- 33%	Completed Tasks to date, as +/- percentage of plan
+ 20%	- %	Hours for Completed Tasks as +/- percentage of plan
+ %	- %	Project £ spent to date as +/- percentage of plan
+ %	- %	Cost of Completed Tasks as +/- percentage of plan

2. PLAN vs ACTUAL	THIS MONTH		TO DATE	
	Plan	Actual	Plan	Actual
Hours			4000	4400
Tasks Completed			174	117
£				
IT headcount				
User headcount				
Other headcount				

3.OUTLOOK TO COMPLETION			
	Original Plan	Latest Approved Plan Date approved: May 1st	Current Outlook
End Date	October 31st	November 30th	December 14th
Tasks	360	390	402
Hours	8000	9000	9250
£	£360K	£405K	£420K

The middle section, section 2, essentially gives the raw numbers from which the percentages in section 1 are derived (we haven't filled in all the numbers).

Section 3, however, is what the sponsor most wants to know. What is Section 3 telling us? We originally thought we'd finish by October 31st. On May 1st, two months ago (let's say it's July 1st today), the sponsor approved a move to November 30th and a budget increase to £405K - that's the current commitment. But as of today, July 1st, our best guess is that we're going to be 2 weeks late and it'll cost £420K which is £15K more than our latest approved budget.

How do you feel about this project? It's slipped once, it's sliding again and the current status would probably make us very suspicious about their projections anyway. And if we felt really nervous we might even want to consider commissioning a Health Check. What's your gut feeling? A, B, C or D? If you *believe* the Current Outlook probably a C. But if, given they are at the moment 33% behind on task completions and completed tasks have cost 20% more than expected and you suspected they had not properly factored that into the outlook, maybe you'd be thinking D. If there is doubt about project status experienced project managers tend to assume the worst case until somebody proves to them it really ain't that bad. Starry eyed, inexperienced young optimists hope for the best, let things run and then get very nasty shocks later in the project.

And how would you feel about the project's change status?

5.PCR STATUS SUMMARY	Hours	£
Spent investigating PCRs to date	200	9,000
Allocated to approved PCRs to date	670	30,150
So, total to date on PCRs COMMITTED & SPENT	870	39,150
In current approved plan, total PCR BUDGET	800	36,000
Total PCR expenditure, estimated OUTLOOK	1500	67,500

number submitted		number approved		number rejected	number under evaluation	Number not looked at
month	to date	month	to date			

						24

There was a budget of £36K for change which has already been overspend and they are outlooking a change spend of more like £67K by the time they finish - they are going to spend almost twice as much on change than they allowed for.

The person reporting then tells you there are 24 change requests sitting on his desk that he hasn't even had time to look at yet. Not good. That D is definitely looking favourite.

A good stage agreement will summarise the plan by declaring a milestone every couple of weeks: start date, requirements workshops finished, sign off achieved, etc.

One project had their plan up on a whiteboard. Every couple of months there was a drawing of a champagne glass. They represented key milestones - if the team met them the sponsor would take them out and buy the drinks. Every few weeks was a drawing of a beer mug. If the team met these milestones they went out and bought themselves a drink. Mineral water, no doubt. You do need something like this to get a bit of team spirit going. Tangible recognition and reward from senior management can also be a great aid to motivation.

5.PCR STATUS SUMMARY	Hours	£
Spent investigating PCRs to date	200	9,000
Allocated to approved PCRs to date	670	30,150
So, total to date on PCRs COMMITTED & SPENT	870	39,150
In current approved plan, total PCR BUDGET	800	36,000
Total PCR expenditure, estimated OUTLOOK	1500	67,500

number submitted		number approved		number rejected	number under evaluation	Number not looked at
month	to date	month	to date			
						24

6.KEY ACTIVITIES LAST MONTH
7.KEY ACTIVITIES NEXT MONTH

8.KEY MILESTONE DATES	Plan	Actual	Outlook

Section 8 shows when we thought we'd achieve the milestones, for those we have achieved when we actually achieved them, and for those still in the future when we now thing we'll achieve them. A quick glance at this section of the report tells us instantly whether the project is more or less running to its schedule.

Page 3 below is more free format: risks, issues, concerns, problems, health check findings: things we've previously reported on and new things we need to bring to the attention of the sponsor.

9.RISKS, ISSUES, CONCERNS, PROBLEMS, HEALTH CHECK REVIEW FINDINGS

Item No: **Month first reported:**

Description of Risk/Issue/Concern etc:

Proposed Action:

Action by whom: **By when:**

Item No: **Month first reported:**

Description of Risk/Issue/Concern etc:

Proposed Action:

Action by whom: **By when:**

Item No: **Month first reported:**

Description of Risk/Issue/Concern etc:

Proposed Action:

Action by whom: **By when:**

For example: "Item 1, sponsor, we declared a major risk at the outset that we might not be able to recruit a graphic design specialist. He joined on Monday, that risk has now gone away. Item 2, we had a health check which recommended that we change our intention to have a single test team and instead have two sub-teams for better control - we'll do it. Item 3, we can't get HR to provide the committed resource to review the requirements document please would you have a word with the HR Director?"

How long would it take somebody to fill in three charts like this each month? An hour or so assuming there is a plan, hours and tasks are tracked, change requests are managed, etc. If planning and tracking are done properly, reporting is quite straightforward. If there's no plan or no tracking, reporting becomes just an unsubstantiated opinion and of little value.

Always do the reporting face to face in presentation mode. If you received those three pages via email, even with tons of backup, you wouldn't know whether the person who sent it to you really understands what is going on. But if they're presenting to you you'll know instantly if they know what's going on in their project.

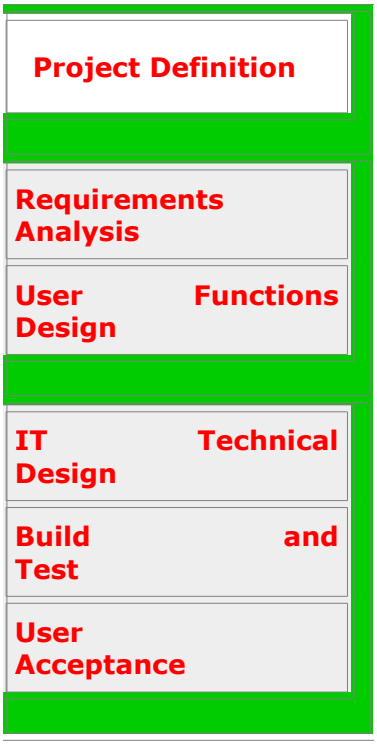
Whether this data is being reported by the team leader on behalf of his (sub) project manager to the project director or by the project manager/director to the project sponsor the same reporting format is used, though the focus of the presentation and discussion may be different. The sponsor won't want too much detail - just reassurance that you'll deliver as committed. The project director will want to dig much deeper. If you were the project director and were shown the data in our illustration above, you would have a hundred and one questions. Indeed, when a team leader is presenting to a project director if he, the team leader, is any good he'll come armed with loads of backup information, such as we saw on the Key Controls report in the planning chapter, and be ready for those questions. If he isn't and can't answer the questions, the project director may not want to wait a month before having another update on that project's status.

In a programme it is useful to have all sub projects reporting to the project director in the same meeting. Healthy competition develops - who's in best shape, who clearly knows what they are talking about. Those whose control isn't quite what it should be will realise they don't know as much about their project's status as their colleagues and will, one hopes, improve. And again, knock on effects between sub projects may become evident in this forum. If a couple of the sub projects are under pressure to meet dates and are contemplating declaring a slippage you may find it becomes a matter of who blinks first. Eventually someone cracks and says they will slip and the others oh-so-reluctantly agree to move their dates later to realign while breathing a silent 'phew!'.

In some IT organisations the senior managers like to see a snapshot each month of all of the projects in which IT is involved - again the same format is used.

Stage authorisation

At the start of each stage the sponsor gives the go ahead and approves the stage budget. Sometimes whether we go or no go isn't in doubt - there's no choice - and it's not so much a meeting to decide go or no go as to demonstrate readiness to begin.



If we're not as ready as we should be the sponsor may give a conditional go: "well, you can start but I'm not satisfied you've got necessary resources properly committed. Get it sorted out and come back and see me in a week's time. And let me know if I can help." We are starting with more risk than we would ideally like, but that's the real world. And as we said earlier, having to present at these meetings is a powerful incentive to get things properly planned.

When we reach the end of Stage 1 we have two things to do: prove we have finished Stage 1 and invite the sponsor to take out his chequebook and fund Stage 2.

For illustration: "Sponsor, we have finished Stage 1. I said it would cost $300K, it actually cost $315K - a bit over budget but well with tolerance. We finished the stage on schedule and we have delivered everything we committed to deliver (UFD, Stage 2 plan, revised business case, testing strategy, etc). We have all the requisite sign offs (here they are...) and there are no issues open on any of the documents we have produced." If that's all true the sponsor says: "Well done, I agree you have finished Stage 1."

But if by contrast the project manager says: "The end date of Stage 1 has arrived. We have published the User Functions Design document and there are currently 99 open issues and everyone has refused to sign off..." Have we finished Stage 1? Certainly not. We must demonstrate that we have finished the work and met the completion criteria specified in the stage agreement.

Without this kind of discipline what happens? Nobody signs off the UFD, work continues regardless and when the system is delivered everyone complains that it's wrong and isn't what they wanted. This is completely unacceptable. If people will not sign off the UFD we must crystallise the issue and deal with it or we must stop work while people make up their minds. The premise being that every day you delay signing off the UFD delays delivery by a day. This is another nuclear deterrent. You hope you never actually have to stop work but the threat must be credible and fully backed by the sponsor. This gives you power to twist arms: either sign off or tell me clearly why you cannot sign off or we will stop work and wait until you make your mind up.

As an example of this. At the end of the design stage of a project the business users, in Finance, just would not get around to reviewing and signing off the design document. After the usual channels had been exhausted the matter was escalated to the IT Director. The IT Director phoned the Finance Director on Friday lunchtime and said: "unless your guys get their fingers out and sign this thing off, or tell us what's wrong with it, as of Monday morning I'm reassigning the technical design team to other work. We're not going to deliver this thing and then have you tell us IT got it

wrong again." This was a fairly brave thing to do because at the time the IT Director reported to the Finance Director! The sign off was forthcoming within the hour.

If we persuade the sponsor that Stage 1 has been finished we then seek authorisation for Stage 2: "Sponsor, four months ago at the beginning of the project we gave you a top down estimate of $1M for the whole project. Having finished Stage 1 and re-estimated we now think the project will cost $1.4M. The benefits are, however, much greater than we first thought so I recommend we continue despite the estimated cost increase."

Why might the sponsor cancel the project at this meeting even if the cost estimate has not increased? The anticipated benefits may have dribbled away - we're better to cut our losses and cancel the project now. We are forcing the sponsor to make a decision: carry on or stop. Without this, some projects simply roll on under their own momentum even though their *raison d'etre* has long since evaporated.

In theory Stage 2 should be authorised before any work is done on Stage 2. In practice the rule might be that you can proceed 2 weeks into Stage 2 before it is formally authorised. In other words there are 2 weeks to sort out any issues and get things signed off. When you're one week and 4 days into Stage 2 the arm twister then becomes: "If you don't sign off by Friday the rules mandate that I must go to the sponsor on Monday morning and ask him to stop the project and redeploy the team onto something else while you make your mind up." Address and resolve issues, don't sweep them under the carpet and have them explode all over you at the end of the project.

Without the go no go meeting with the sponsor - the summit signing ceremony - it is much easier to fudge these things, drift from one stage to the next and leave UXBs (unexploded bombs) all over the place.

Definitions

Checkpoints and milestones are not the same thing. A milestone denotes an achievement: a workshop has been held, a document has been published, a document has been signed off - the last of these being the best sort of milestone as it signifies real achievement. Publishing something is easy, getting it signed off is much more significant.

A checkpoint is a point at which one checks things, e.g. a status meeting in which one checks whether milestones have been achieved or not.

Original plan, baseline plan, latest plan, latest approved plan, current plan... It's amazing how many names people will invent for various versions of the project plan. One way of using some of these terms might be:

- Original plan. The plan you started the stage with. By definition this plan can't change. At the outset, since there is only one version of the plan, the original plan, baseline plan, latest approved plan, etc are one and the same thing
- Latest approved plan. If during the stage the sponsor approves a change to the stage's commitments (revised budget, different dates, etc) the plan that

supports these new commitments is the latest approved plan. Some would call this the baseline plan or the new baseline plan or the latest baseline plan.
- Latest plan. The project manager and team leaders will have a working level plan that shows when individual work tasks will be done. This plan can be changed quite substantially without affecting commitments made to the sponsor. This 'latest plan' is the one against which the project team is tracking its progress. If they achieve it they will achieve commitments made to the sponsor.

If you speak to ten project managers you will probably get eleven different opinions of the above, but whatever you call them there will usually be several versions of 'the plan' scudding about. As long as you're clear what they all are, call them what you like - though consistent naming within your company would obviously be a good idea.

Objective, factual reporting

Objective, factual, numerical reporting makes it much harder for the project manager to hide problems and much harder for senior managers to ignore painful reality.

Having to present status face to face means the project manager has to know what's going on in his project.

All (large) projects should report using the same reporting template - senior managers soon tune in to the language of the numbers.

Most businesses employ less than 10 people but must report numerically in their annual report and accounts. Many projects employ more than 10 people. Look upon these larger projects as a small business: they require the same sort of entrepreneurial flair, leadership and control that a small business requires.

Before your project begins you must decide what you will track and how you will control. For a small project a Gantt chart on a whiteboard with the plan as green bars and the actual as red bars may be all you need. For large projects you will need something of the sort we have described. For very large and complex projects you may well require sophisticated critical path analysis software to guide you through the maze. Decide what's appropriate for your project, design controls accordingly and allow time in the plan for gathering, analysing and reporting status data.

In a large project that does not have any factual status reporting data the following does, and will continue to, happen:

- On Friday afternoon the IT team prove to the IT team leader that the project is dead in the water, and it is, but there are no numbers to prove it.
- On Monday morning the IT team leader tells the IT project manager: "there are major problems, we aren't going to make it", but there are no numbers to prove it.

- On Tuesday morning the IT project manager tells the project director: "there are some big problems, it looks rather doubtful", but there are no numbers to prove it.
- On Wednesday morning the project director tells the project sponsor: "IT have some problems, but they always say that don't they?".
- On Thursday morning the sponsor tells the Board of Directors: "it's business as usual, it'll probably be OK."

It's gone from being dead as a dodo to probably OK because each person changes the message a bit on its way up. But if even the sponsor when reporting to the Board must show the sort of numbers we have seen, that can't happen. The Board can read the numbers as well as we can - probably a lot better than we can. However high the reporting goes some numbers should be presented. Just like company accounts.

And finally

- You cannot control unless you have a plan
- Good control detects problems early
- Early detection makes resolution easier
- If in doubt assume the worst case until something better is proven
- With facts it's harder to hide the truth and harder to ignore the reality

Chapter 11 - Change and Issue Management

At the start of a project there is always uncertainty. However well the requirements are defined, they will change as the project progresses. Issues will arise that were not foreseen.

What is an 'Issue' in the context of a project? A risk that has happened, something unexpected that has occurred, a potential problem that has come to light - in other words a threat to project success has been identified. We need to deal with these threats, not leave them lying around festering.

What is the difference between an Issue and a Risk? Some people will offer definitions and distinctions and argue their view with religious fervour. In practice what sometimes happens is this. Threats to project success that are identified at the start of the stage or project are called risks. Things that crop up during the project are handled as issues - though if a potential problem that wouldn't bite for 3 months crops up maybe that should be added to the risk register and handled as a risk whereas if the problem would bite in two days time maybe that's an issue. You can see how the arguments start.

Frankly it doesn't matter too much whether these things that crop up during the project are called issues or risks. What does matter is that they are managed.

In a small project the issue management process may boil down to discussing the issues in the weekly status meeting, assigning them to a person to resolve and documenting in the minutes the issue and its eventual resolution.

In one very large project the project office devised an electronic process for handling issues. There were 6 categories. Category 6 could be raised within a team and resolved by the team with nobody above the team leader being involved. But if a category1 issue was raised, as soon as enter was pressed an email went to the project sponsor - by definition category 1 issues required the sponsor to make a decision. The system produced excellent reports of issues raised by category, number open, number overdue resolution, average time to close, etc.

Whichever extreme your project is closer to the issue management process comes down to this:

- Before the project starts design an issue process that will meet your project's need.
- When an issue arises write down what it is.
- Assign the issue to somebody to resolve. In a small project the project manager will decide who to give the issue to for resolution. In a large project that might be delegated to the team leaders - they decide which team member should be asked to resolve an issue.
- The person asked to resolve the issue does whatever it takes to get it resolved.
- Escalation paths must be open. The last thing you want to happen when the first major issue arises is not to know how to get to the sponsor - so you send him an email which sits in his inbox for a week. Before the project starts, agree with the sponsor and steering committee members how you'll get urgent issues to them - via their secretary, via a phone call? Get them to agree they will make timely decisions. (In one case this ground work hadn't been done. A key team member left, the PM identified a contractor with the same rare skills. An email to the sponsor asked for authorisation for the cost. It sat in the sponsor's inbox for a fortnight. When he eventually said yes the contractor had got a job elsewhere.)
- Write down how the issue was resolved.
- In status meetings monitor open issues and give particular attention to issues that are past their target resolution date. Occasionally what seems to be a minor issue is raised, given to a team member to resolve by Friday - but they don't resolve it by Friday - and what seemed to be a minor issue conceals a much bigger problem lurking underneath.

Let's have a look at a sample issue form and illustrate the process with an example. Half way through the build phase one of the developers is about to write some code that will do a complex interest rate calculation. The calculation specified in the design document looks wrong to him. He's not quite sure so he raises it as an issue and completes the form as far as the 'Description of Issue' box.

ISSUE FORM	
Project Name:	**Issue No:**
Short Description of Issue:	
Raised By	**Date Raised:**

Description of Issue:	
Resolution Comments:	
Closed By:	**Date Closed:**
Issue Closing Agreed By:	**Date:**
No. of PCR (if one was raised as a result of closing this Issue):	

Someone - maybe him maybe someone else - asks the relevant business person whether the calculation specified is right or not. If you're lucky the business person says that although it looks odd it is in fact perfectly correct. The resolution is a huge sigh of relief - there's no problem.

If you're unlucky the business person says the calculation has got completely garbled somehow and is completely wrong. The expert spends five minutes writing down the correct calculation. However, this represents a change to the signed off User Functions Design document and as such it must be managed as a Project Change Request. So the issue resolution is: the calculation is wrong in the UFD, this is the correct calculation, PCR number 735 has been raised.

A third way of closing issues is also possible. For example, the team is growing and was due to move in to the refurbished canteen next week. But the refurbishment isn't finished - we have an issue. After investigation the conclusion is the

refurbishment will be 3 weeks late and in the interim the team will colonise the sponsor's palatial office - that's how the issue will be resolved. You probably wouldn't raise a PCR for that.

So issues can have 3 closings:

- there's no problem - phew.
- there is a problem but it has been sorted out like this...
- there is a problem which necessitates a change to a previously signed off project deliverable: the issue is closed, a PCR is raised and the change is managed under the auspices of the change control process.

One thing to watch out for on larger projects: make sure the person who raised the issue is satisfied it is closed. Otherwise, if the loop isn't closed like that, and they aren't happy with the closing they will raise the issue again under a slightly different title.

The form above is only an illustration - as are any forms to be found in project management standards or in a methodology text book. Before your project begins design a form that meets the specific needs of your project.

Project change requests (PCR)

A project change request is a request during the life of a project to change a signed off project deliverable. So, once the PDD has been signed off, if later in the project someone wants to change the project scope they raise a PCR. Once the User Functions Design document is signed off if someone wants a change to the design they raise a PCR. We are not discussing here changes that are requested to live systems - we would call these Improvement Requests. However, if an improvement request is raised and a live system is changed and there happens at the same time to be a project underway to enhance that system, we may also need to raise a PCR to reflect the change to the live system in the copy of the code the project developers are working on.

What happens if continuous, unbridled change is allowed during a project? The project will go on for a very long time and may never finish. At the other extreme allowing no change at all is a nice idea but would probably mean that whatever was delivered wouldn't meet the business need.

Almost always in a project some change has to be taken on board as the project progresses. Clearly we need a process for managing that change. A key aspect of this process is deciding, at the start of the project, who will be authorised to spend the company's hard-earned cash on changes. We will consider later who this should be.

How will the PCRs be administered? If only a handful of PCRs are expected the project manager can administer the PCRs himself. But some projects, notoriously software projects, get hundreds or even thousands of PCRs in which case a project office will be needed to handle the administration. For medium sized projects perhaps a team leader can look after the paperwork.

Planning for PCRs is like quadruple crystal ball gazing:

- how many PCRs might be raised during the stage
- when might they be raised
- what type of changes might be requested
- and therefore who would investigate and do the changes

Putting the change hours in the right team members' plans at the right time is very difficult, but the change hours must be put somewhere in the plan. If in doubt put the change time at the end of the stage plan as we discussed in the planning chapter.

Never muddle the change budget with contingency. There should be a clear, separate budget for change with a clear owner. Contingency is there for anything else that might crop up. Though one reason for spending contingency could be that there is more change than expected and contingency is raided to fund it.

Of course, one could begin the project or stage with a zero change budget. Every change that is accepted increases the project budget and, in principle, moves the project end date. However, this will almost certainly be perceived as failure.

Setting aside 5%-15% of the stage budget for change is typical and may be broken down into three pots:

- a number of hours for administration of PCRs
- a number of hours for investigation of PCRs
- a number of hours for making changes

A project might spend more time investigating PCRs than actually making changes. Why might this be? Having worked out what a change would cost everyone decides it's not worth doing. In a way this is a waste of money - investigating a PCR that then gets rejected. We will see how we might limit this wasted expenditure.

A change request process typically has three steps.

In the first box (please have a look at the form below) the person raising the PCR says two things: what change they would like and why - what is the business justification for disrupting this project half way through; who raised the PCR; and when was it raised.

The PCR then goes to somebody - say the IT team leader - whose first reaction to a PCR is usually a sharp sucking-in of breath: "tch, oooh, just to investigate this change request would take me two days." We now have a 14 hour estimate just to investigate the impact of this PCR (the first line of box 2).

Somebody should now have to make a decision: do you want to spend 14 hours worth of the company's hard earned cash to investigate this PCR or not? At the extremes PCRs come in two sorts. At one extreme everyone know the PCR has to be done so we get on and do the impact analysis. At the other extreme, particularly in software projects, there can be hundreds of 'bright idea' change requests. An extreme example of that: we are half way through the build step and some bright

spark thinks it would be a nice idea if one particular web page, rather than being green on black, flashed on and off bright pink but only on February 29th every leap year. Just a bit of fun for our customers.

This great idea goes to IT who say it would take a month just to establish if that was feasible, let alone the cost of doing it. At this point even the bright spark who had the idea would say: "forget it!". The point is that one can save wasting limited PCR investigation time by saying what the investigation would cost.

Project Change Request (PCR)	
Project:	**PCR No:**
Description of Proposed Change and Business Reason:	
Submitted By:	**Date:**

Estimated Hours to Investigate Change Request:		
Accept Reject For Investigation	**Project User Manager**	**Date:**
Accept Reject For Investigation	**Project IT Manager**	**Date:**
Reason for Rejection: (add attachments if necessary)		

Actual Hours Spent Investigating Change Request:

Description and Impact of Change: (add attachments if necessary)					
Estimated Implementation Cost By Phase:	Current phase	Phase:	Phase:	Phase:	Total Hrs: £:
Estimate Valid Until: (delay will increase change cost)					
Approve Disapprove For Implementation:	Project User Manager				Date:
Approve Disapprove For Implementation:	Project IT Manager				Date:

If we decide to investigate the PCR, someone establishes what the change would cost and what its impact would be on the project's schedule. Let's say (box 3) it takes 12 hours to do this investigation. That's 12 hours spent from the change budget, we can't spend it again.

Project Change Request (PCR)	
Project:	PCR No:
Description of Proposed Change and Business Reason:	
Submitted By:	Date:

Estimated Hours to Investigate Change Request:		
Accept Reject For Investigation	Project User Manager	Date:
Accept Reject For Investigation	Project IT Manager	Date:

Reason for Rejection: (add attachments if necessary)

Actual Hours Spent Investigating Change Request:					
Description and Impact of Change: (add attachments if necessary)					
Estimated Implementation Cost By Phase:	Current phase	Phase:	Phase:	Phase:	Total Hrs: £:
Estimate Valid Until: (delay will increase change cost)					
Approve Disapprove For Implementation:	Project User Manager				Date:
Approve Disapprove For Implementation:	Project IT Manager				Date:

And, for example, the investigation reveals that the impact of the PCR would be a 2 week delay to the project end date; 100 hours (£5000) extra work in the current step (let's say we're in IT technical design), 50 hours extra cost in the build and test step (another £2500) and 80 hours extra user testing time (£3000). So total cost 230 hours (£10500).

Very simply someone can now make an informed, business-like decision: does the benefit of this change justify the date hit and the extra cost (and justify jeopardising quality - see the next chapter). That's all the PCR process is trying to do: let the paying customer - the user - make informed decisions about change.

On the form (again, any form is just an example to help you devise one appropriate for your project) you'll notice the 'Estimate Valid Until' box. Imagine you're having a house designed and built for you. The design has a 2 car garage. You ask what it would cost to change it to a 3 car garage. The builder says it depends: if you make your mind up by the end of the week it will be £1000. But next week the foundations are being dug and concrete poured, so the £1000 estimate is only valid until the end of Friday - leave it later and it will cost you an ever increasing amount.

On a large project this approach can galvanise people into making timely decisions on change: make your mind up by Friday or it will cost you more than this. This

helps prevent a large number of change requests sitting around awaiting decisions, each of which is a potential source of project disruption.

Where do change requests come from? If you're not careful they will assail you from all directions: the users change their minds (for good and bad reasons), the IT guys have great ideas for 'improving' the design, company policy changes, unforeseen external events affect the project. But unfortunately in software projects most change requests result from two causes:

- errors and omissions in the requirements
- vague, incomplete or just plumb wrong design documents

When running a software project we would recommend that each change that is taken on board is categorised by cause. If you discover that most change is caused by vague requirements that will drive you almost subconsciously to invest more on the next project in getting the requirements right: rigorous, hothouse requirements definition workshops, etc.

If you were a non-technical person managing an IT project and one day a techie said: "boss, please can you sign off this PCR. We want to change the name of this object from XYZ-3 to XYZ/3. The cost is virtually zero, there is no impact on the user's view of what we'll deliver." Would you really want to see that change request? You wouldn't even know what they were talking. But if changes of that sort are not carefully managed amongst the IT team there could be chaos.

So you might invite the IT team leader to run his own cut down PCR process for these small technical changes. But be careful: this can become a back door change control process through which things get slipped that should have come to your attention as project manager. Sit down with the IT team leader every now and again and get him to talk you through the changes on the small change log and woe betide him if you understand any of them.

Clearly, at the other end of the spectrum there could be a PCR that would cost more than the whole change budget. Obviously the sponsor would need to approve it as it would increase the project cost.

Before you begin you might decide that any PCR that would cost more than 25% of the change budget will not come out of the PCR budget, they will be an addition to the project budget - the PCR pot will be used only for changes smaller than that. Indeed, you might draw a matrix that shows for each type of change who has to be involved in investigation and consideration of it and for each size of PCR who has to be involved in approving it. So from the outset you know that a technical change costing less than £x can be handled by the IT team leader, but a business change costing more than £y needs a decision by the sponsor. Work out the rules of engagement before each stage begins: the rules will probably be different for each stage and may even change during a stage.

Estimating what a change might cost can be very tricky. Back to that house you are having built for you. At the design stage you want a window moving 3 feet to the left. The change won't cost much and it is very clear to the architect what the cost will be - he's just got to change the drawings.

However, when the house is nearly finished how much will that change cost? A lot of money - that's obvious. What is less obvious is this: the builder thinks he'll send someone with a large hammer to knock some bricks out of the wall, move the window across and brick up the other side - to you £1000. But when he's knocking the bricks out of the wall what does he discover? A water pipe or a power cable, so he has to get the plumber or electrician in. The change costs a lot more than he estimated. If he has given you a fixed price quote that's his tough luck, but in IT projects the user ends up paying one way or another whatever IT say the change will cost. People tend badly to underestimate the cost of changes that come along later in the project.

It is worth recording the estimated cost of each change and recording what each change actually cost and then working out by how much PCRs at each stage tend to be underestimated. These wobble factors can then be used as fairly crude adjusters next time around. So next time, you estimate a change during build will cost £2000 but you know you usually miss things and changes at this stage typically cost 50% more than estimated, so you quote £3000 as the estimated cost to try and give the user a more honest view of the likely true cost of the change.

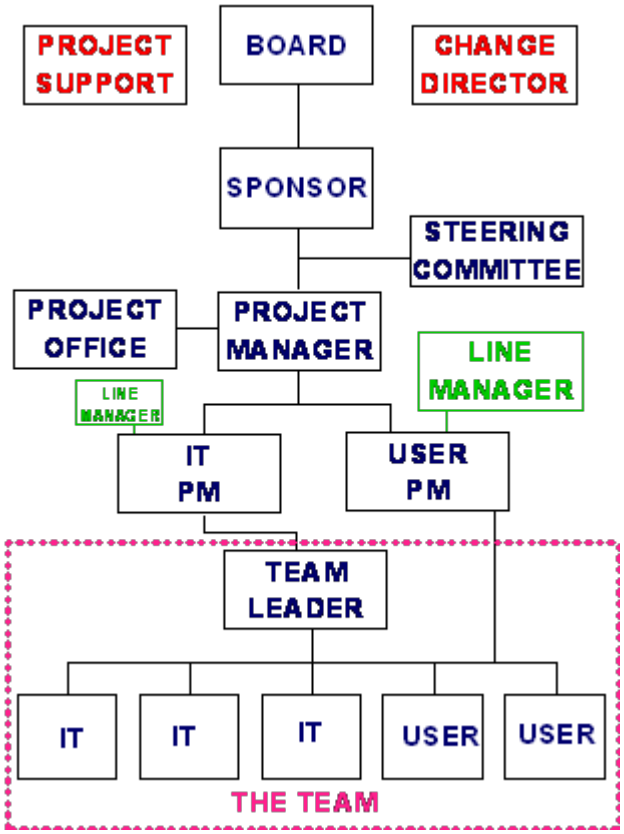
The later the change comes in the more it will cost and the more it will tend to be underestimated, as illustrated by the graph. Beware of taking on changes during the later stages of testing - one small slip can cause significant delay and the change can cost many many times more than expected.

If you're managing a project where development is outsourced to a software house and you want a change that the software house knows you have to have, how much can the software house charge you for that change? Anything they like.

This should be a powerful incentive to get the requirements and the UFD right in order to avoid these potentially huge change costs. A fixed price contract is only a fixed priced contract if you don't change your mind. If you do keep changing your mind it will be the fixed price plus a very large amount of money. In the next chapter we will address how we might try and get it right at the requirements and design stages.

A software house can make a lot of money on change requests, so one might argue that maximising change is in their interest. Not necessarily so. There can be so much change that the project gets out of control and fails - that does neither client nor supplier any good. And even if the project eventually delivers, the client may feel they have been taken to the cleaners on change and go elsewhere in the future.

Being everyone's friend and saying yes to change is easy. Saying no isn't quite so easy. In a project some change will be essential to the user/customer/client. From the project manager's point of view all change is bad news: it is disruptive, tends to introduce errors and can have unforeseen consequences. Minimising change is in just about everybody's interest. But if we go back to our roles chart, which of our generic role players should be the first line of defence against non-essential change?



The Project User Manager. On a well set up project it might work like this. The sponsor says: "I understand we need to set aside 10% of the project budget for essential changes. Project manager, please come and account to me at the end of the project how you spent my change budget and woe betide you if you wasted it on flashing-on-and-off-pink type change requests." The project manager will exhort the project user manager saying: "you are closest to the real needs of the business, only allow through changes that really are essential."

A good user project manager is worth his weight in gold. He knows how to say the all important word: "no". Politely but firmly: "no" - unless it is essential or has an overwhelming business benefit. A good defence against potentially valuable but not absolutely essential PCRs is to make them candidates for a future release. The PCR is closed and an Improvement Request (or whatever you call such things) is raised which will subsequently compete for inclusion in a future release.

A good user project manager won't even let people raise a PCR form for trivial changes as they clutter up the works and cost money to administer, let alone to investigate.

For internal company projects approving change requests does not feel like spending real money, and user reps can end up approving change like there's no tomorrow.

In one case a large project had a project user manager and 5 business areas whose requirements would be met by the system under development. The project user manager and 5 user reps were the first level change control decision makers. Hitherto they had been approving any change that anyone felt was desirable and the project was being engulfed by a tidal wave of changes and sinking fast. A new process was instigated. At the first meeting the user project manager said: "who would like this change request included in the project?" Five hands went up. "Under the new process I must tell you that it will add £10,000 cost to the project, do you still want it?" Again 5 hands went up. "And under the new process I have to ask you: whose departmental budget is the £10,000 going on to?" No hands went up - nobody wanted it *that* much. Find some way of making it clear to those approving change that it is real money they are spending.

In weekly team meetings mention every change that has been approved in the past week. Someone may pipe up: "don't you realise that affects all the training materials

I've developed?" And you didn't. Make sure the team know about all accepted changes.

Change can cause major difficulties if you're not careful. In one case on a commercial project the software supplier said they would charge to investigate PCRs unless the investigation would cost less than a man day, in which case the investigation would be done free. Good idea?

The project ran into difficulties. The project manager was replaced. The new project manager found 4,000 PCRs sitting on the backlog awaiting investigation, each of which it was reckoned would cost less than a day to investigate. What would you have done? Apparently the new project manager went to the customer and showed them a list on his laptop of the 4,000 PCRs. He said he was contractually obliged to investigate them for free but he said: "I'm not going to." And he then dramatically pressed the delete key and they all disappeared. He said if any were really important re-raise them - and sue if you want to - but otherwise they no longer exist.

Brave man! But everyone knew they were just so much clutter and were pleased to see that at last someone was prepared to take a hard line on this and other matters and get a grip on the project.

Are errors found in system test PCRs? If the system is not performing as per the user functions design that is an error - it is not a change request. If the system is performing as per the UFD but someone wants the system to do something different that is a PCR - even if the UFD is plain wrong it's still a PCR. The developers are contracted to deliver whatever it says in the UFD: if you want something else delivered that's a change and it will cost you. So get the UFD right. See the next chapter.

Finally,

- Be realistic, you'll always get some change
- Include a change budget in the overall project budget
- Make someone accountable for how the change budget is spent
- Design Issue and Change Control procedures that meet the needs of your project
- Good control is forward looking. Always ask: how much have we spent so far on change and therefore how much more do we think we will need to spend on change. If it becomes clear half way through that the change budget is nothing like big enough get more change budget *now* from the sponsor, don't wait till it's all been spent.
- Minimising change in a project is in just about everyone's interest.

Chapter 12 - Quality Management

Which of these would be the better way to do a project:

- Do it right as quickly as possible
- Do it quickly as right as time allows

This represents a choice between two quite different approaches to projects. Setting out to do it quickly as right as time allows not only means there is a perfect excuse for not bothering too much about quality, it will probably mean the project will actually take longer: driving for quality tends to reduce project cost rather than increase it. Set out to do it right as quickly as that is possible - that should mean the project delivers better quality *and* gets done more quickly.

We trust you do not recognise any of these symptoms of poor quality in systems:

- errors in live running
- system crashes
- floods of improvement requests
- unhappy users
- overtime during testing
- mushiness

Mushiness? Go and look at a project and everything about it is mushy. Nothing seems to be firm, pinned down or disciplined. It can be an indication that quality, amongst other things, isn't being managed.

What causes errors in systems?

- continuously changing requirements
- no clear "manufacturing process", no-one is quite sure what should be done at each stage
- lack of testing
- no process for removing errors
- and many more

Even in some large projects there is no process in place for getting rid of errors as the project progresses and then everyone wonders why there are huge numbers left to be found at the end - surprise surprise.

And some project teams never ask the question: what causes us to make errors? If that question is never asked, error causes are unlikely to be addressed and the same problems are repeated over and over, but nobody realises.

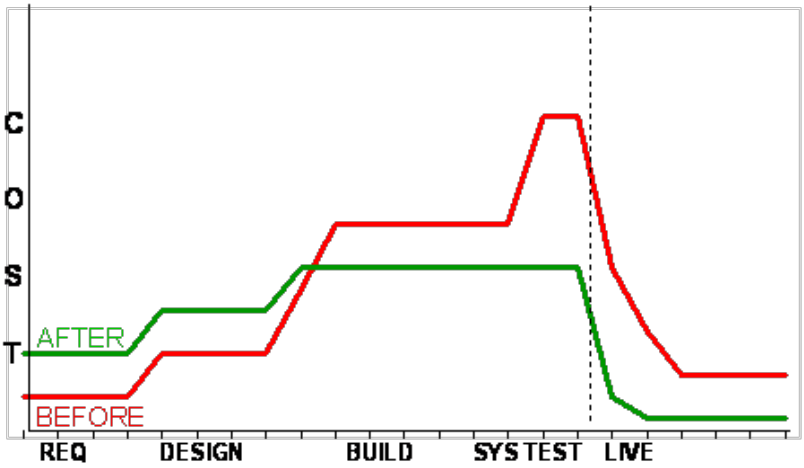
When building anything complicated the errors probably won't be distributed evenly throughout it, they will tend to cluster together. If the project manager and the team have no idea where these clusters are, corrective action will not be focused where it is most needed.

So often people say they want to deliver good quality but in practice on the ground they are doing little or nothing to achieve it. As project managers we need to manage quality not just hope for it.

What is quality in the project context? Delivering what the customer/user asked for in terms of functionality, auditability, enhanceability, performance, etc and meeting the requirement to deliver on time and on budget. If the customer wants a bicycle we deliver a bicycle that works not a gold-plated Rolls-Royce. Adding unnecessary features is not adding quality it's wasting money.

If the customer wants a system with 1000 bugs in it and we deliver one with 1000 bugs in it we have met the customer's requirement - a quality job has been done. If the customer, by implication, wanted a system with few or no bugs and we deliver one with 1000 bugs we have failed. And that is usually what the customer wants - a system with few or zero bugs. Unless the customer states otherwise, this should be the target. Can we guarantee zero defects in software? On the contrary, we can be almost certain there will be some defects - we can almost guarantee we will fail to meet the zero defect target. This may sound odd, having a target we know we will fail to meet. We will resolve this apparent conflict later in this chapter.

Quality should not be something we hope for vaguely and we should not only measure it after delivery - that's rather too late. Quality should be something that has objective, even numerical, targets set at the outset of the project, a plan for achieving those targets and some way of knowing as we go along whether we are meeting those quality targets. Quality should be as planned, predicted and measured as any other aspect of the project - something we manage actively rather than hope for passively.



Getting the Quality Message

The chart shows two project cost profiles. One before getting the quality message, the other after seeing the light.

Before getting the quality message a relatively small amount of resource is spent getting the requirements roughly right but if we're honest not really correct or complete. The user functions design is sort of right but again not complete or correct and not fully understood or agreed by the users. Then during the build stage all the problems start to crawl out of the woodwork. And sometimes there is a huge panic peak in testing where a very large number of errors are tested out and the actual user requirements are tested in. And when the system goes live a flood of bugs to fix and enhancements to be made.

When people get the quality message they invest a lot more during the requirements step: rigorous, hothouse sessions to identify business processes fully and in detail and then time and money to check that the requirements are right. At the UFD stage it is very hard to over-invest in getting it right. Churning out good quality code from

a thorough, complete and correct UFD document is relatively straightforward these days. And then no panic in testing and many fewer problems when the system goes live.

Looking at the chart, which will be the cheaper way of doing the project? The 'After' way, the quality way: what you invest up front in getting the requirements and design right you will almost certainly more than get back in build and test - and the real benefit comes when the system goes live.

Unfortunately the notion that going for quality will make the project cost less is not how it feels at the beginning: all people perceive is this 'enormous' cost of being meticulous and of checking, checking, checking. This cost is real and tangible and certain whereas the savings that will come in the build and test step are only potential. The easy option is not to fight to incur that upfront cost, and instead to fire-fight the 'inevitable' problems at the end of the project. Be strong, plan those upfront checks in - you may find nobody tells you to take them out: it's your own inhibitions you're fighting, not others' resistance.

When talking to business users and sponsors about quality IT people have a tendency to say things like: "We should improve quality, we are getting far too many defects, far too many bugs, far too many crashes." And the users have no idea what you're talking about - they don't know what these words mean and so they aren't interested. How might we interest the sponsor in quality? Talk about money.

Some bugs found in live running are cheaply and easily fixed. Others can be very expensive indeed, not so much to IT to fix, but expensive for the business. There is a nice story of a UK bank that sent a letter one Christmas to its richest clients thanking them for being good customers and entrusting the bank with their money. The programmers, as programmers always do, made up silly names when testing the production of the letters and due to a slight oversight several thousand of the bank's best customers got a letter which began: "Dear Rich Bastard". Apparently some were highly flattered, but others were not at all amused and took their money elsewhere.

We are all aiming for no errors in live running - they can have expensive consequences.

Finding errors in system test and user acceptance test isn't quite as expensive but is still expensive and the more errors there are the longer testing will take and the more it will cost. Ideally there would be no errors to be found in system test or user test.

If programmers can find errors when testing units of code it's cheaper but certainly isn't cheap - ideally there would be no errors to be found in unit testing.

Finding errors by inspecting requirements and design documents is very cost effective but certainly isn't free. Ideally there would be no errors in these documents.

The only ultimate aim a quality programme can have is to end up with perfection: no error is ever made at any stage of any project. Now of course we will never reach that promised land. But each time we do a project we should try and get a bit closer,

and project by project try to drive the errors out earlier in the lifecycle - the earlier they're found the cheaper they'll be to fix.

This implies we need a plan: how will we find errors early, how will we know we're succeeding, how will we reduce the causes of errors?

Quality planning

In the PDD or stage agreement, under the heading 'quality plan' we could say: 'in this project we will not check anything, we will not do any testing, the system will go live completely untried and the errors will be found by our customers.' That is a perfectly valid quality plan. It's not a very good one, but it is a very clear statement of what we intend to do, or rather not to do, quality-wise.

But we hope you'd be saying something more like this: 'we will check the requirements and UFD documents like this... we will analyse error causes like this... we'll measure quality like this... and these are the team's quality related targets...'.

In a small project deciding what should be checked by whom is not too difficult. In a large project deciding who should check what and building quality checking tasks into the plan can be complicated, which is why some project managers have a specialist - we'll call him the quality leader - inside the team (we stress inside the team) who helps decide what level of checking is appropriate, builds quality tasks into the plan and makes sure the team execute the quality tasks. More of this role later.

Would you agree that the sooner we can find an error the cheaper it will be to fix it? Then perhaps we should measure the time gap between when errors are introduced and when they are found and if it's on average a short gap financially reward the project team, and if it's a long gap (and all the errors are found near or after the end of the project) financially penalise the team.

Quality in all projects

Within a company it could be that some projects deliver excellent quality while others deliver terrible quality. The chances are that those producing good quality are checking up front, the others aren't.

How might we get all projects to include appropriate quality checking activities in their plan? Or should we leave it to project managers and hope they plan quality tasks in - and let the cowboys and the inexperienced learn the hard way? One way would be to have a rule that at the start of each stage the project manager must review his plan with project support, as we suggested in that chapter, and project support must agree appropriate quality tasks have been built into the plan. (We stress appropriate. In a simple project done by experts very little checking will be needed. In a complex project staffed by novices a lot of very thorough checking will be required.) Again, this is not about showing project support some boilerplate document called 'Quality Plan', in means project support reviewing the task by task work plans to assess whether in their judgement quality tasks are properly planned in. And a further rule might be that the sponsor will not give the go ahead unless project support are satisfied. In this arena policing is sometimes required to bring the laggards on board.

Error cost in business case

How about this: at the start of a project the project manager must estimate, perhaps based upon past projects, how many errors might be found in live running. He must multiply by an average cost per error (i.e. roughly what it costs the organisation - business costs and IT costs - each time a bug is found in a live system) and include the resulting amount of money as a cost item in the project's business case.

Imagine you're the sponsor and the project manager presents his cost breakdown to you and says: "...and based upon past projects we expect 50 errors in live running at an average cost to fix of £10,000, so the bug fixing cost will be around £500K." What would your reaction be? You may well ask the project manager what he could do to reduce that cost of failure. Suddenly the sponsor is interested in quality. The project manager than describes how he could make that more like 20 errors and £200K by investing up front in getting it right. And you will need the sponsor's support for this because much of that up front cost will fall upon his people - the users, as we shall see.

Team quality briefing

It is a good idea to have a kick off meeting at the beginning of each project stage. Invite the sponsor along to say a few well-chosen words, including these: "Team - user people and IT people - I don't really know what program bugs are but I do know that each one you deliver to me will cost me something like £10,000 to sort out, so I don't want too many of them please."

The team know the sponsor is interested in quality, not just any old thing delivered on the right date. The ground is now fertile for the project manager to run through the quality plan: "This is how we're going to try and meet the sponsor's requirement: we are going to check our work like this... we are going to measure and analyse errors like this... we are going to analyse and address error causes like this... and, team members, these are your individual quality objectives upon which your next appraisal will partly depend." The team know the sponsor wants good quality and also that it is in their personal interest to deliver good quality. We will discuss team members' targets later in this chapter.

Inspections

An inspection is a meeting to identify errors and omissions in a piece of work. One cannot sit down and inspect a 1000 page document in one meeting. Twenty pages is about as much as can be done in one inspection. So that's 5 inspection meetings if you're producing a 100 page document and 50 inspections if you're producing a 1000 page document. Already you're thinking it's too expensive - hold on.

We do not wait until all 1000 pages are written before inspecting. Each section is inspected as soon as it is produced. Because the team are essentially checking each other's work, inspections are also a great aid to understanding what's going on elsewhere in the project.

The inspection is attended by the author, a moderator to chair the meeting and a number of inspectors - we'll see in a moment who they might be.

Suppose you have written 20 page section of a user functions design document. You send it to the moderator and the inspectors. Each inspector will have, say, a 2 hour task in their plan to check, on their own, the section for errors and omissions.

There is then a meeting - another task in each attendee's work plan. The moderator asks those present: "how long did you spend preparing for this inspection?" If they all say "no time at all" the inspection is cancelled. If you were the moderator and someone said they had spent 3 hours preparing but you knew they were fibbing and they had spent no time at all, what would you do? You might complement them and ask them if they would quickly run through all the errors they found. They won't try that again. The moderator has to make sure inspections are done properly, we're not just going through the motions.

Sometimes moderators will invite someone to read the material out loud. Other times it might be a case of: "Paragraph one any comments? Paragraph two..?" Whichever approach is taken we all know where we are. If someone pipes up and says: "the part number is missing from this web page layout" and we all agree that it should be there, fine. But if a long debate threatens to break out the moderator guillotines it. This is a meeting to record errors and possible errors, not a meeting to have long discussions about how things should be corrected.

The author makes corrections after the meeting and, strictly speaking, the moderator should check that they have. Again, this correction task will be in the author's work plan as we saw in the planning chapter. Occasionally there are so many errors that major rework is needed and the moderator decides a re-inspection is required. One hopes this does not happen too often but clearly the overall plan has to allow for a certain percentage of re-inspections.

The moderator records the total cost of the inspection - preparation hours plus meeting hours - and the number of errors found. Only real errors are counted. Generally, if the error would have resulted in a bug in the final system it is counted as an error. Spelling mistakes in internal documents that will not have any subsequent effect on the project might be noted for correction but not counted in the statistics. How might we use this data in the future? If we discovered from this simple measurement that it typically costs about 2 hours work to find and correct errors by inspecting the UFD and we also know it takes about 10 hours work to find and correct similar errors in system test - in other words it's five times cheaper to find and fix errors via UFD inspection than via system testing - that should drive us almost subconsciously to invest more in UFD inspections next time. But if you never measure this you'll never know which is the cheapest way of finding errors and you won't even have the evidence to persuade yourself to invest at the right time, let alone the evidence to persuade anyone else.

In principle any project deliverable can be inspected but the biggest bang for the buck will usually be had by inspecting requirements and UFD documents.

When checking the UFD one is largely checking that it satisfies the relevant bit of the requirements, so have a copy of the requirements document to hand when preparing

for and holding the UFD inspection. Similarly, when inspecting the IT technical design have the UFD to hand.

We will see that driving for zero defects does not imply infinite cost nor unconstrained cost, nor even great cost - nor actually *any* cost. The tasks that we put into the plan for inspection preparation, inspection meetings and re-work following inspections are like any other tasks in the plan: they have an allotted number of hours. And the original task to produce the bit of work will have had a number of hours within it for the author to check his own work. We are not pursuing quality regardless of cost - we are investing a predetermined amount of effort and money in what we believe to be the most cost effective way of removing errors. Further, this investment should be recouped via lower testing costs later in the project - and the real 'profit' comes because there will be fewer problems when the system goes live.

Can't we rely upon people to get things right without all this checking? If only we could. Defining a complicated business process fully and unambiguously is very difficult. Producing complete and correct systems designs is very difficult. Checking will be needed. And by the way this does not let the author off the hook - he still has an obligation and an incentive to get it as right as he can, as we shall see towards the end of this chapter. In the process-oriented world the aim is for people to get it right without checking. If we found they weren't getting it right we would change the process (or change the people). But if Finance was producing the Annual Report and Accounts or a Takeover document there is no way one person, however skilled, would produce that and it would be published without being checked. Inspections are not unique to IT projects. They are used when anything complicated, and usually one off, is being produced where correctness is important.

If we were inspecting the heart of a complex system design we might need many people there to verify its correctness from their perspective - inspections with a dozen people present are not unknown. But if by contrast an expert has done something very simple a dozen people checking it might be a bit over the top. There are three equally valid forms of inspections:

1. The full works with a meeting as described above.
2. A multi check. The author writes something, sends it to a couple of inspectors, they check (again, they will have tasks in their plan to do this), they email any errors to the author but there is no meeting.
3. A simple check. One person has a quick look at the author's work.

The real fine art of quality planning is this: at the beginning of the project stage thinking through every bit of stuff that will be produced and deciding whether it will need 12 people in an inspection meeting to check it or a quick over-the-shoulder-glance check, and then planning all those inspecting tasks into the project plan. This can be horrendously complicated to do. This is why it helps to have a quality expert in the team who can take the lead in this.

For example, let's say we are planning the project on the right. Who should attend the inspection of the section of the UFD indicated?

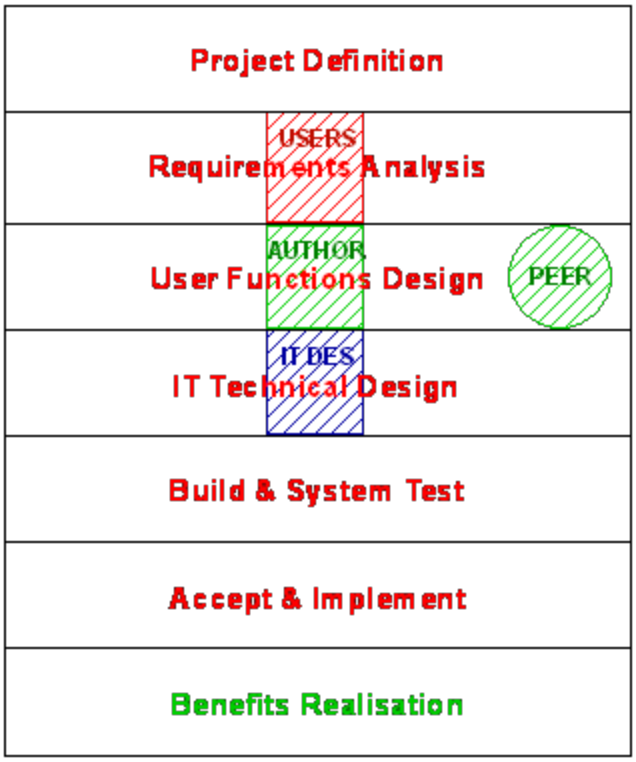
First, the author must be there. Also the 'user', though this is actually two people: the person who wrote the relevant bit of the requirements document (who may never actually use the system) plus someone from the genuine end user community who will eventually use this part of the system. And what should those two users say at the end of the inspection? It meets the requirements and we are happy to use it as it has been designed. When in the bad old days did the users give their verdict on those two things? Perhaps in user acceptance testing or even when the system had gone live. A bit late in the day.

Think back to that house you are having built for you. The architect rings to say he's designed the kitchen so you go into his office, check the design and sign it off - or point out how you'd like it changed. One week later he rings to say he's designed the bathroom. You go in and check it out. If it was your money and your house would you do those design inspections thoroughly and carefully? You bet. Unfortunately in IT projects the users don't have quite the same incentive to do the inspections. Whose job is it to make them do the inspections properly? You guessed it, the project manager.

Again we stress that these inspections are tasks planned into the users' project plans. At the start of the stage their time was committed to the project manager by their line managers and the individuals have agreed to perform these tasks.

Who else should be at the inspection of that section of the UFD and why will this next person probably be the best inspector of all? The person who must do the relevant bit of the IT technical design. He will be a very assiduous inspector because his ability to do the IT technical design depends upon the UFD section being complete correct and intelligible.

And who else, later in the project, will be working from this section of the UFD? The test team, so we may want someone there from the test team to verify they will be able to base their testing upon it. Incidentally, involving the test team in this way can also result in the design being changed so that the system will be easier to test, which can save time and money later in the project.



We may also invite other members of the UFD design team to the inspection to check that this bit of the design fits with their bit. You can see that quite big teams can form to check key sections of the UFD.

Generalising, an inspection team comprises:

- the author
- the moderator
- the author(s) of the previous stage to ensure their work has been translated correctly
- peers - people alongside the author to check for correctness and for fit with their bit
- the author(s) of the next stage plus any others who will use the thing being inspected
- experts who we think will be able to help us find errors

Moderators do need to be trained (takes half a day) to ensure inspections don't descend into personal abuse, arguments over grammar or discussion about document prettiness. We are after a short sharp meeting that identifies things that, if left uncorrected, would have been bugs in the final system. The moderator will need to be a strong chairman to curtail debate about how errors should be corrected.

One person who should not attend inspections is the project manager. If the team *like* the person whose work is being inspected no errors will be found in the meeting (and after the meeting the author will be slipped the list of errors under the table). If the author isn't so popular all the errors will be painfully dragged out in front of the boss. Make this a pure peer review process without the beady eye of the boss skewing it.

A word of caution about multi checks. They can seem like a cheaper version of inspections but can end up costing more. Suppose you have written a section of a UFD. You send it to the relevant users to check it. They send errors and comments back to you. You correct the UFD and sent it back to them. Someone who was happy with the first version now thinks there is something wrong in the revised version. You do your best to change it to their satisfaction and re-issue it, but now someone who made no comments at all on the first version makes loads of comments on version three - maybe because they didn't read versions one and two. And so it goes on, ping-ponging back and forth. Particularly where many people are reviewing a piece of work it can be cheaper and quicker to hold an inspection meeting.

Plan inspection preparation tasks, inspections meeting tasks and re-work tasks like any other project tasks and then resist the temptation to skip them when the project gets behind schedule.

Inspections can be mind-numbingly boring so try and give inspectors some incentive to put their backs into it. One simple idea is to put a chart up on the wall which plots how many errors each team member has found (note: found not made) and reward this month's top inspector with a £50 award. This is an unequal competition as some people will attend more inspections than others but at least it provides some recognition for those who are finding errors and saving you and the team stress later in the project.

Inspections are good value for money - but you will need to measure this for yourself before you'll really be persuaded. However, they do have one major weakness. If a team sits down to inspect a section of a document they tend to check what is there - they don't always spot what is missing.

Simulations

There is a technique we will call simulation that can be very effective at finding the gaps that inspections miss.

Having finished the inspections of the UFD, and believing it to be correct, gather the team in a room and open the UFD at page 1 which, say, shows the initial web page the customer will see. Generate some test data, e.g. post code X74Y-4, and invite someone to be the computer. He slavishly, without adding any of his own knowledge, executes the logic specified in the UFD and says: "ah, if post code X74Y-4 is entered this pop up would appear saying invalid post code." We may even have the screen layout on a transparency and write in X74Y-4 then show an OHP of the pop-up. This low-tech approach works well.

Then try test case number 2. This time it's a valid post code and the simulator tells us which web page would then be displayed to the user. Perhaps someone else now takes over the role of the computer and some more test data is 'entered' to see what the system would do. The simulator says that the data would be stored: post code P74 7YQ; Mr Smith; 74 Arcadia Avenue; Part number 34712; quantity 3. Someone takes on the role of keeper of the database and writes this data onto a flip chart on the wall.

When we get to page 392 of the UFD and we are trying to produce an invoice - actually doing the calculations with a calculator using data on the flip charts - the person doing it says: "oops, I can't calculate the invoice amount because the customer discount code, which gives the discount percentage, is blank". And we realise the discount code has to be a mandatory field rather than an optional field on the input screen on page 94. That kind of error tends otherwise to remain dormant until the later stages of system testing.

You will usually find that the simulation has to be stopped after the first hour or so. Why? The design just doesn't say what's supposed to happen next. So you stop the simulation, plug the gap and come back to it. It can take a couple of weeks to simulate right through a system but that could save a month of system testing. Give it a try, you will be amazed what a simulation will find.

This has been described as getting the users to do the user acceptance testing at the design stage: it's much cheaper to find the errors here than later when you have a million lines of code to unpick and rework. The simulation uses test data which can

be re-used during system test. Simulations are excellent education for the test team. And those charged with writing user guides will find participating in simulations very informative.

How does this differ from prototyping? Prototyping is often interpreted as showing users a series of screen shots to illustrate how the dialogues flow. A simulation is much more than that. We are pumping test data through the system logic to see if it produces the right outputs. We are exercising not only the visible parts of the system but the invisible parts as well. We are system testing the UFD.

If you're thinking there's no way your design documents are ever at a level of detail or state of completeness that would allow them to be simulated in this way you have such a fundamental problem that no quality assurance techniques in the world are ever really going to help you. Simulation is a technique for refining silk purses, not a technique for the miraculous transformation of sow's ears. If you haven't got the basics of the software manufacturing process right everything we are describing in this chapter will sink without trace into the quagmire of your disorder.

Simulations require the project manager to make a decision: to do simulations rather than not do them. Getting the bugs out by system testing doesn't require any decision to be made: if there are bugs you will have no choice but to keep testing. Simulations represent a pro-active approach to quality management, leaving it all to testing a reactive one. Strangely, people will sometimes tell you they can't afford to have someone spend two weeks doing a simulation but they will happily commit that same person for two months to do testing - nobody will argue with the need to throw in limitless resources to sort out the manifest problems in testing - there is no choice. Earn your money, be proactive, get the bugs out early, save money.

If the requirements document is inspected and simulated (yes, the business processes can be simulated too) and the user functions design document is inspected and simulated and the IT guys inspect their IT technical design and do code inspections (a.k.a. walkthroughs or code reviews) there will be many fewer errors left to be found in system and user testing.

If you have no personal experience that leads you to *know* that inspecting and simulating up front will mean many fewer errors in system test you may be reluctant to plan significantly less testing time than you normally would. That's OK. Plan in inspections and simulations *and* the usual amount of testing time - belt and braces. Then when system test finishes early because you have run out of bugs to find nobody will complain. But this might mean 'quality' shows up as an addition to project cost in the estimates, even thought the actual outcome will be a reduced project cost (everything else being equal!).

Expected results

As soon as the user functions design document has been signed off, the system test team can start work. They can produce test data and work through the UFD to determine the expected results. For example, the customer orders 3 of product x at £n and 4 of product y at £m - the tester can then work out what data should appear on the invoice by following the logic specified in the UFD. But the tester realises that the expected result is very obviously wrong. He has found a bug months before he has done any actual testing, maybe even before the code is written.

When he comes to do the testing it should be quick - all his thinking has been done in advance - he just has to check the system produces what he was expecting, he doesn't have to spend time working out whether the system has produced the right result.

System testing

If the UFD is complete and correct how much business knowledge would one need to do system testing? None at all. The best testers can read and write but have no other obvious skills. There is only one way they can test - by laboriously generating 'what if' test cases based upon the UFD. After all, system test is fundamentally checking one thing: does the system do what the UFD says it should. And the very best testers of all are university vacation students - they're bright people, they work hard because they want a job when they leave university and they get paid peanuts - it's like employing slave labour to do testing.

In practice a good test team comprises a test team leader (or test team manager in a large project), the junior people to do the donkey work plus one or two wise heads to say if the system is *really* right despite what the UFD says. Leading system test is a highly skilled job - in a moment we will discuss when the system test leader should join the project team.

In projects that are modifying packages or bolting new functions onto them the UFD will be the design for the changes or additions to the package and the same principles will apply in system test: do the changes and addition perform as per the UFD. If it is a tried and tested package we will not need to test its underlying functionality. However, if we are the first users of a package or a new version of a package we may want to test it as well as our changes: this can mean the test cost is disproportionately high when compared to the build and unit test cost - something to watch out for if we are using rules of thumb for design cost vs build cost vs system cost, etc.

Is the following reasonable: the developers get behind schedule, skip the planned quality checks, deliver late to system test, system test overruns because there are so many bugs, the end date slips and the system test leader/manager gets the blame. Not very reasonable.

With good quality management the following becomes possible. At the start of development the test manager and the development manager agree how many errors the developers will deliver to system test to find - let's say 300. (The prediction at the simplest could be based upon how many bugs per development man month have been found in the past in system test). They further agree this would mean that around 50 bugs would be found in the first week of system testing and further still that if more than 80 are found in the first week that will constitute statistical proof there are many more than 300 bugs in the system. And the system test team will go home and the system will be handed back to the developers until they can show they've done their job properly. This is another nuclear deterrent, we hope we never have to use it but everyone on the production line has the right to expect that good quality work will be passed to them. Even if, when it comes to the crunch, you decide not to suspend system test it must be made clear that the development manager has failed and he should be appraised accordingly. If he

knows at the outset he will be held accountable for the quality of what he delivers he is more likely to take quality seriously.

Having completed system test without the usual panic and maybe even early, the system is thrown open for a final user acceptance test - which should be something of a non-event. After all the users know, having done the simulation, that the system is acceptable and they should find very few, if any, errors in this final test. Indeed this should be more a demonstration of how wonderful the system is than a test.

When is a bug a bug? If the system is not performing as per the UFD that's a bug. If the system is performing as per the UFD but the UFD is wrong and you want it changed that's not a bug, that's a change request.

If quality management has been good during development we will know roughly how many bugs will be found in testing: we can plan the number of tests to be done per week, the expected number of bugs per week, the expected number of bugs outstanding at the end of each week, the average time to fix bugs, and then we can track actual performance against that plan. And one could devise several other measures to track both the quality of the system and the efficiency with which bugs are being fixed. A key measure is the bug finding trend: if the number being found each week is above plan *and* keeps going up each week the alarm bells start to ring. A declining number of bugs being found each week may indicate we're on the home straight.

A lack of quality management earlier in the project will mean system test is unpredictable and the project manager is not so much managing it as clinging to a tiger's tail. Manage quality throughout and system test becomes as predictable and manageable as any other part of the project.

Quality Measurements

Why might we want to measure quality? If we want to find out whether it's cheaper to find errors by simulation or by system testing we'll need to measure the cost effectiveness of each. And everyone already uses *some* quality measurements - we all count the number of bugs found in testing, don't we?

A lot of quality professionals go around muttering strange statistical terms like "standard deviation", "least squares fit" or "six sigma" which nobody else understands. If one is not careful the whole business of 'quality' becomes something far too complicated and esoteric for most people. (One organisation appointed a quality manager. Everyone expressed support - that was the right thing to be seen to do. He started to collect charts from projects and soon his office wall was full of them. Quality then became a matter of who could produce a chart so pretty that it would make it onto the wall and displace a rival's. They all had copies of the software which could produce lovely zigzag plots bounded by control lines, elegant bell shaped curves - great fun. Nobody had any idea what the charts meant and nobody used them for anything, but they seemed to make the quality manager happy.)

The purpose of quality measurement is not to produce pretty charts and statistically elegant metrics, it is to stimulate and direct quality improvement. We prefer simple raw numbers the average team member can understand that are used by the team to do help them deliver better quality outputs.

One could devise hundreds of ways of measuring some aspect or other of quality - we will show just a couple to illustrate the principles.

Imagine your project team have just finished inspecting and simulating their business requirements document. The team report to you the number of errors they found in each of the document's five sections. What would you ask the team?

Requirements Inspections and Simulations

Section	Errors	Pages	Errors per 10 pages
change account type	16	3	53
close account	24	20	12
open account	43	45	10
change interest rate	25	39	6
data transfer	2	3	6

"Why were there so many errors in the 'change account type' section?" The chart shows there were 16 errors in 3 pages which is 5.3 per page (though they chose to show errors per 10 pages to avoid decimal points). That's four times more errors per page than any other section. You would want to know what that was. And then what would your follow on question be? "What are you going to do about it?" If you can ask those two questions - "why's it like that?", "what are you going to do about it?" - you can manage anything. We have found an error cluster, we can address the problem. The team say: "we did have a problem with that section, the user who wrote it didn't really have the right knowledge - it is currently being revised by a more experienced person and it should not be a running sore throughout the project." The measurements are beginning to enable us to manage quality.

A couple of months later the team show you the results of the user functions design document inspections. What would you be asking about this time?

User Functions Design Inspections and Simulations

Section	Errors	Pages	Errors per 10 pages
data conversion	36	12	30
change account type	18	18	10
interest rate change	29	52	6
close account	15	39	4
open account	24	63	4

The data conversion section of the UFD tops the poll so you'd certainly ask about that. Maybe the data transfer requirements, only 3 pages long, had been a bit vague and high level and only now at the design stage are the problems coming to light? Would you ask about anything else?

Too slow, missed your chance. A month or two later you get the results of the IT technical design inspections. What would really be worrying you now?

IT Technical Design Inspections

Section	Errors	Pages	Errors per 100 pages
change account type	39	63	62
close account	17	110	15
data conversion	7	57	12
open account	115	124	9
interest rate change	2	39	5

Change account type again. What are we being told? When the IT team inspected their technical design they found four times more errors in the change account type section than in any other part of the document. We'd want to know why.

If you're lucky the team say: "it was done by a trainee, he made a bit of a hash of it - not really his fault - but it has already been reworked and is now up to scratch." That would be good news. Bad news would be: "boss, the requirements for 'change account type' keep changing - it's chaotic. We need to go back and redo the requirements for 'change account type' and re-inspect them, redo the UFD and re-inspect it then completely redo the technical design for 'change account type' from scratch and re-inspect it."

Will that be painful for the project manager? It will, but a lot less painful than discovering the problem three months later in system test and then having to do all that rework plus re-write the code, delay testing and who knows what else. To be a good project manager it helps if you're a masochist: you are willing and able voluntarily to inflict pain upon yourself - with the aim of avoiding twice the pain being inflicted upon you involuntarily later. (In our example it would seem the team didn't spot the problem at the requirements stage - though perhaps they did but chose to ignore it or they failed to sort it out properly. One hopes they would at least have dealt with it at the technical design stage.)

The quality leader would show the project manager numbers like these each week or month. But he wouldn't just be showing the numbers - he'd be saying something like: "boss, we realised from the numbers there was a bit of a problem in this area and we have put John Spencer on it and he will have it revised and corrected by Friday." In other words the team, prompted by the quality leader, are using the measurements to help them identify and fix any quality weaknesses as the project progresses.

One more example of a measurement. You've just finished the project shown in the table below and you're about to start the next similar project. On the just finished project the team found 3 errors when they inspected the PDD and 41 when they inspected the requirements document. When they inspected the user functions design document they found 120 errors but they realised 3 of them had in fact been

errors in the requirements, which had obviously been missed at the requirements inspection.

(Look at the column headed 'UFD': 120 is the number of errors found at the UFD inspections and the number 3 just above is the number the team judged were errors in the requirements document. For example, the requirements said 'invoice amount is quantity x discount%' and the UFD faithfully reproduced that 'invoice amount is quantity x discount%' but at the UFD inspection it dawned it should actually say 'invoice amount is quantity x price x (100-discount%)'. Though few errors are quite that obvious!)

	Step in which error found							
Step in which error caused	**PDD**	**Req**	**UFD**	**ITD**	**B&UT**	**Sys**	**Live**	**%**
PDD	**3**	**0**	**0**	**0**	**0**	**0**	**0**	**100%**
Requirements		**41**	**3**	**2**	**1**	**0**	**2**	**84%**
User Functions Design			**120**	**4**	**9**	**7**	**1**	**85%**
IT Technical Design				**185**	**49**	**41**	**0**	**67%**
Build & Unit Test					**292**	**68**	**11**	**79%**
System & Accept Test						**141**	**15**	**90%**
Live Running							**15**	

The IT team found 185 errors in the technical design and even 2 of those were in fact requirements errors that had been missed both at the requirements inspections and the UFD inspections.

If you wait long enough you even can put errors found in live running on the chart. And 2 of the errors found in production were errors in the requirements which had got right through all the nets.

Is there any particular weakness on the production line quality-wise that you would like your team to address before the next project begins?

At the right hand side of the IT technical design row there a figure of 67% - what's that telling us? When the IT team checked their technical design they found two thirds of the errors, and one third of the errors that were under their noses in black and white, they missed. Is that good, bad or criminally appalling? It's dreadful. And the project manager should say so and invite the team to fix it before the next project. If it transpired, say, that the IT technical design hadn't been well structured which made it difficult to check it all fitted together properly, the project manager might insist that before the next project the team figure out how to construct the IT technical design so that it *can* be cross checked properly, and he would be looking for an efficiency nearer 90% next time.

Identify which step in your process is weakest and improve it - and keep doing that for ever.

You may be thinking that the measurements we have looked have taken no account of the severity of the errors nor the cost to fix them, and you'd be right - you are starting to devise some of those hundreds of ways of measuring quality-related things. The danger though is that one gets carried away with devising ever more

'correct' measurements. A crude, even not very accurate measurement that drives improvement is infinitely preferable to a statistically perfect one that doesn't.

We mentioned a couple of pages ago the sort of measurements the system test team will use: errors found per week (probably broken down by severity), number open, average time to fix, etc. Extrapolating from these numbers after the first few days or weeks of system test should enable good prediction of how long testing is going to take and how many errors will be found in total.

And really good quality management will enable the following sort of discussion. Suppose at the beginning of the project we try to get commitment of user resources to do thorough user functions design inspections, but for apparently good business reasons they cannot be made available. We will be able to predict how many extra errors this will result in both in system test and in live running. We can then say to the sponsor: "by not investing this £30K's worth of user effort in UFD inspections there will be about 20 extra bugs in the system when it goes live which will cost you about £200K to sort out - what would you like to do?" Now, the sponsor might decide the current business need is overriding and he'll pay the failure cost. Or he might decide the current business need isn't *that* important and direct the manager of those users to make them available to get the UFD right. Quality becomes a thing we can have a business-like discussion about rather than something abstract that is hard to get a hold of. But we are approaching a PhD level of sophistication here which, regrettably, all too few software projects aspire to let alone achieve.

Cause analysis

In the best run projects someone, perhaps the quality leader, gets the team together every couple of weeks to ask: what errors have we made in our work in the last two weeks, what caused us to make them, what can we do to remove or reduce those error causes? This is not about finding someone to blame for the errors but we do need to know who has made them: for example if the contractors are all making the same mistakes over and over perhaps some training is needed to address the cause. The aim is to identify what we can do to eliminate the causes of errors that we as a team are making. These meeting are sometimes called quality circles.

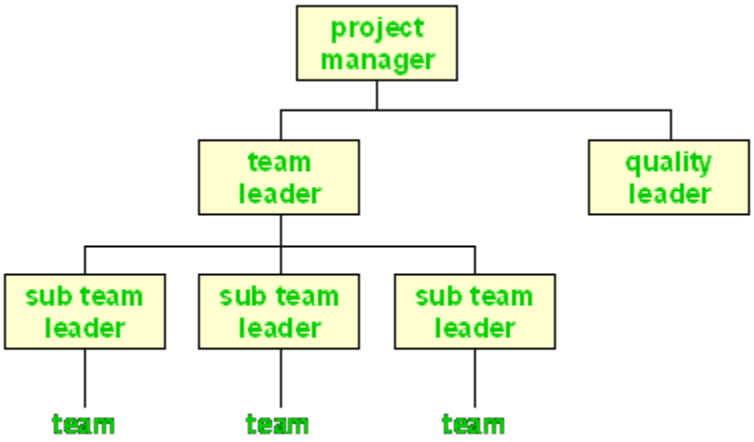
Usually when one starts to run these causal analysis meetings some very 'obvious' error causes are identified which can be fixed quite easily, and everyone wonders why they weren't sorted out ages ago. Sort them out now.

Quality lessons learned review

At the end of the stage a two hour team quality lessons learned meeting is held. The team look back over the whole stage and identify the top two or three causes of errors and work out what could be done to engineer those error causes out of the project process in future. Again, try to avoid heaping all the blame on people who aren't there, and focus on what *you* can do to improve *your* part of the process. Though if most of the problems experienced in the stage were in fact caused upstream in an earlier stage the team may want to develop constructive proposals to take to those who were involved in that earlier stage. Primarily, though, the team should be identifying what they can do to remove error causes before the next stage or the next project begins. Equally, when these meetings are first held some quick

wins can usually be had - things that can easily be changed that will significantly improve the initial quality of work and/or the timely and effective detection and correction of errors.

Some would strongly advise that the project manager should not be present at causal analysis meetings or these end of stage quality lessons learned meetings as this can inhibit soul baring.



Quality leader

Let us round out the role of quality leader:

- must be a member of the project team, not an outsider
- may be a part time role perhaps for one of the team leaders, but will be a full time role on a large software project
- help build the quality plan: who will inspect which bits of others' work and when
- ensure the team perform the planned quality tasks properly
- organise inspections, acts as moderator for some inspections (though any team member can moderate inspections)
- organise and drive simulations
- compile and analyse quality measurements and ensure the team address quality problems revealed by the measurements
- report on quality matters to the project manager
- chair causal analysis meetings and ensure the team take action to eliminate error causes
- run stage end quality lessons learned meetings and ensure action is taken to improve quality management in future stages and projects

Even if there is a quality leader in the project team he does not become accountable for the quality of what is delivered. Each author remains accountable for the quality of the work they produce. With one exception to that rule. If, heaven forbid, the quality at the end of your project is awful and you, the project manager, conclude the reason is that the inspections had not been done properly then you shoot the quality leader! It was his job to make sure the team did the inspections properly and if they weren't to come and tell you about it.

But imagine this. You are the quality leader of a software project that your company is doing for a client. The team leaders are refusing point blank to let their team

members do the planned inspections. Why? Because they are behind schedule. What would you do?

You go and speak to the project manager and point out that the way things are going the quality is going to be terrible. But the project manager says: "never mind about that, we've got a date to meet for the client." What would you do now?

You speak to the project manager's boss, say you're being overruled and that quality checks are being skipped. And you show the statistical evidence that quality is going to be poor.

Let's say this triggers an independent health check and they conclude you are right, it's going to be a disaster in quality terms the way things are going. The senior manager removes the project manager and the new project manager tells the client he is going to delay delivery by two months to ensure a good quality system is delivered.

The point of the story (and by the way any similarity to real events, characters or projects is purely coincidental) is that the quality leader has a real responsibility to protect both the supplier and the client from the consequences of poor quality.

In that example it was the supplier who, belatedly, dealt with the problem and sorted it out before delivery - the client was fortunate.

Suppose you are the client sponsor of a software project. When it's delivered and installed your company's business will depend upon that system working properly. If the quality is really bad it could even put you out of business. Who would you make accountable for the quality of what was delivered to you? You might appoint a project manager in your company to manage the project and the relationship with the software supplier. You would make him, your in-house project manager, accountable for the quality of the system the software supplier delivers to you. Now put yourself in the position of that project manager. You're accountable for the quality delivered by the supplier! What would you do?

Well, you'd make very sure the supplier had good quality management procedures in place - you may even commission independent consultants to conduct an audit of the supplier's quality processes if you didn't have the expertise to do it yourself. You might contractually oblige the supplier to put a quality leader in place and you might oblige them to present to you each month the sort of quality measurements we have seen in this chapter. Do you really want to see the numbers? Not really. What you do want is evidence that they, the supplier, are managing quality as they go along. You may even want them to predict and commit to, on pain of penalties, the maximum number of errors they will deliver in the final system.

Let us bear in mind also that even if you are outsourcing a software development project you're not outsourcing the whole project - only the easy bit, the technical, design build and test. The bit that will have the most influence on the final quality - the defining of the requirements and reviewing and approving the user functions design - will be your in-house responsibility. Only your people, the users, can define the requirement; only your people can verify that the system design is correct. You, the client project manager, have got a very powerful incentive to ensure that your company invests users' time up front and in things like inspections and simulations if

you will be accountable for the final product meeting the business need and having few or no errors.

When should the quality leader be appointed? In an in-house project, where the project manager is managing the project from start to finish, the quality leader should be one of the first appointments that is made - the quality leader should be in place right from the beginning of the project. It needs to be made clear to him that it is his responsibility to ensure the team do whatever it takes to make the project's deliverables good quality. The quality leader should also be told that half way through the project - at the end of the UFD step - his job title will change and he will become the system test leader/manager. Why should this work like a charm? If the quality is bad come system test, who will spend those nice warm summer evenings and weekends in the office during system test? The system test leader. This should give him a powerful incentive to ensure the team really do get the bugs out at the requirements and UFD stages.

Quality leaders tend to select themselves - they have a passion, even an obsession about quality (a good thing!) about things being right - and they drive the rest of the team accordingly.

How might we appraise the quality leader - how do we know if they have done a good job? If the team find 50 errors when checking the UFD and 50 more come to light later in the project, was a good job done at the UFD inspections and simulations? No, only 50% of the errors were found. Blame the quality leader. If by contrast the team found 90 at the inspections and simulations and only 10 came to light later, the quality leader did a great job in getting the team to check things properly. And if 95% of the errors are found at any stage that's pretty spectacular - give the quality leader a pay rise.

Similarly, if system test finds 50 bugs and 50 are found in live running how good a job was done by the test team? Abysmal. If testing finds 95%? Wow.

There is a risk that people will start building errors in so that they can be found by inspection and make the numbers look good. This shouldn't happen but it may be worth gently pointing out to the team that since performance appraisals and possibly salary increases partly depend upon these metrics, fiddling the figures would constitute fraud.

With really good quality management in place it becomes possible to establish benchmarks. Let's say we typically find 20 errors per 100 pages at UFD inspections. That's the benchmark. If next time around the team find 50 errors what would your suspicion be? Maybe they're getting sloppy and submitting half-baked work to inspection (or building in errors so that they can be found - not that they would do that, of course...). And if next time they only find 2 errors per 100 pages what would you suspect? They are not doing the inspections properly. You can investigate and take action - you're really beginning to manage quality.

Team members' quality objectives

"Team member, every error found in your work post inspection will count against you in your next appraisal. But the meter only starts running *after* your work has been inspected." Why might this work really well?

It means that authors become agents for quality. They demand a good inspection of their work - errors found there don't count against them. But perhaps you're thinking people will therefore submit unchecked work to inspection and let their colleagues find all the errors for them? Only once. It is embarrassing to be in a meeting at which your work is being inspected and it is riddled with errors - it's obvious you haven't done your job properly. In really bad cases the moderator may stop the inspection and tell the author to sort the work out and resubmit it when it's in a fit state - which can be quite humbling. Next time one hopes the author will do what he can to get it right - and glow with pride when few or no errors are found at the inspection.

We said earlier the target should be zero defects even though we know we'll fail to achieve it. If the target is any other number - say 20 and there are 10 team members - they all know they're allowed 2 errors each. They won't check their work all that well on the basis that a couple of errors are OK. And there will probably be 5 or 10 or 15 errors there actually.

The team leader's target is also zero: if a member of his team is continually submitting bad work to inspection that jeopardises the team leader's appraisal too. The team leader will, one hopes, offer support to that team member to help him improve the quality of his work. If there are several development teams and there is a certain amount of rivalry over which team will have the fewest errors found in their work in system test, that team member may also come under some peer pressure to pull his socks up. Generally, the more errors there are in a piece of work prior to inspection the more errors will be missed by the inspection, so anything that can be done to help and encourage authors to get it right themselves will have the biggest influence on the final quality. And when we reach the promised land inspections and simulations just verify that there are indeed no errors there.

Fundamentally we are trying to reduce the number of bugs in live running. If in the past there have typically been say, 20 bugs in live running for every 50 man months of project effort, and this time there are only 5 errors per 50 man months, you know the team have done a good job. There may be other factors such as project complexity and team experience to take into account, but that's the final criterion - did we deliver a good quality system and save a lot of problems and cost in live running. That's the only reason for attending to quality - to save money.

If you are brave you might even want to try this. At the beginning of your next project explain to your team all about inspections and simulations. Tell them you are a quality fanatic. Tell them you'd like them to plan the project for you and tell them they can plan in as much checking time as they like at the requirements and UFD stages. There's a risk they'll take you to the cleaners but they probably won't - even the *prospect* of *too* many inspections can result in brain death.

But by saying they have *carte blanche* to plan is as much time as they like in order to get things right at those early stages how have you almost guaranteed a good quality outcome? You have removed *the* excuse for bad quality: we didn't have enough time. The team know that one won't wash with you so they will check their work that one more time just to make sure it's right.

Consider these two projects:

Project A. The perception is that the schedule is very tight, there isn't enough time to do quality checks in the requirements or design steps. The requirements and design steps 'finish' on schedule but the documents aren't really finished and are in fact riddled with errors. Build is fraught and runs behind schedule. Testing takes a very long time and delivery is late.

Project B. The perception is that the schedule isn't *too* tight. Quality checks are planned in at the early stages and are done, and the requirements and UFD are top notch - as complete and correct as one could hope for. Build runs like clockwork. Testing goes to plan and delivery is on time.

Project A and project B are the same project.

Objective quality and subjective quality

When the system goes live, counting the number of bugs (by severity), the number of crashes and outages and the number of mandatory improvements gives us an objective measure of the system's quality. However, users sometimes love a system despite all the bugs - it makes their lives easier, does what they want, and even has some sexy features. Conversely they may hate a system that is available 24/7 and has no bugs at all. So it's worth doing a user satisfaction survey to gauge this subjective view of the system's 'quality' - more on this in the next chapter.

And finally:

- Everyone in a project team should have a written quality objective upon which they know their next pay rise partly depends.
- Set out with the aim of getting it right first time every time but have a secondary aim of finding any errors as soon as possible after they are made.
- Build good quality in. No manufacturer ultimately succeeds by throwing bad quality things together and then trying to test out the bad quality at the end of the production line.
- Plan quality checking tasks into the project plan. It may *feel* expensive at the outset but will make the project cost less in the end: rework *is* expensive.
- Identify error causes and take action to reduce them.
- There is only one reason for driving for quality. It saves money. Crosby said quality is free - it's better than that!

Chapter 13 - Stage and Project End

And now, the end is near... and no doubt you did it your way. But could others learn from your travelling of each and every highway?

There are three ends to be neatly wrapped up:

- stage end
- project end
- benefit assessment period end

At the end of each project stage

Let's see if you have been paying attention: when you produce a deliverable such as a user functions design document how do you know who has to sign it off?

At the start of the stage you agreed who would sign off the stage deliverables and you documented that in the stage agreement. People reviewing and signing things off should be subject to the same project disciplines as any member of the project team: if they agree to sign off within two weeks of publication, the project manager should make sure they do. And if they agreed at the start of the stage they'd sign something off it's more likely they will be involved during the stage and as a result one hopes there will be fewer problems and surprises at the end.

Team lessons learned meeting

Sometimes known as a team *post mortem*, hold a team lessons learned meeting at the end of each stage. Typically this is done the week after the 'real work' of the stage has finished, in the tidy up period. It may only be a 2 hour meeting but put it in as a task in each person's plan otherwise it may not happen.

Some say the project manager should not attend. We want the team to confess their sins – the project manager's presence might inhibit that - so the team leader may be the right person to run the meeting. The whole team should be represented or preferably present. Ask them to come prepared to articulate what went wrong and how similar problems could be avoided next time. But also have them think about what they did really well and how we might ensure those successes are repeated next time.

The chairman will need to be careful that the meeting does not become a witch hunt – spending 2 hours vilifying an absent third party may be fun but doesn't really help. It's all about what *we* can do better next time. Emerge from the meeting with an action plan lest the good intentions evaporate.

An example. The team conclude that a lot of time was wasted during the design stage because they didn't have a clear idea at the outset how the UFD document would be structured. Someone is actioned to put together a template detailing what should go into each UFD chapter in future projects.

If the project manager isn't present at the meeting the team leader would run through the conclusions with him.

It is important to do this at the end of each project stage: if you leave it and only do it once at the very end of the whole project you will long since have forgotten what happened in the early stages.

Some advocate you shouldn't even leave it unil the end of the stage: have a lessons learned log that is continually updated as you go along as well as having the post mortem meeting at the end of the stage.

Share experience

Your natural generosity of spirit might incline you, as project manager, to share your experience with other project managers simply by going and talking to them or by making a presentation at the next project managers' forum (if you have such things).

However, project managers sometimes won't do this, and there are a couple of reasons why. The project went extremely smoothly because the project manager applied what, to him, are very obvious, common sense disciplines – so obvious, in fact, there's no point in talking about them. But of course to a novice those things aren't at all obvious. At the other extreme, the project was such a painful experience the project manager doesn't even want to think about it, let alone go around talking about it.

Either way valuable experience doesn't get shared. Which is why there is often a rule in place that at the end of each stage key data and lessons learned must be captured in a stage end report.

Stage end report

The stage end report records:

- which stage the report is about
- what kind of project it was and what technology was used
- how many people (at the peak) worked on the stage
- who the key role players were
- original planned cost and end date of the stage
- extra budget approved by the sponsor
- actual cost and end date
- tools & techniques used and would you recommend them
- things that helped and how to ensure they are repeated
- things that hindered and how to avoid them
- how the company's project management rules and guidelines could be improved

And for example the information could be recorded on a template something like this:

Stage End Report (Page 1)	
Project Name: **Stage:**	**Project Number:** **Risk Assessment:**

PROJECT TYPE	TECHNOLOGY
( ) Normal Lifecycle ( ) Rapid Application Development ( ) Package Modification ( ) Maintenance Release ( ) Other:	( ) Java ( ) VB ( ) C++ ( ) Package: ( ) Other:

Number of Individuals Involved: IT TEAM:	**IT OTHER:** **USERS:**
Project Manager:	**Dept:**
Project User Manager:	**Dept:**
Project IT Manager:	**Dept:**
Project Leader:	**Dept:**
Project Sponsor:	**Dept:**
Brief Description of Stage:	
Which SER is this? (First/Interim/Last/Cancelled/Overall) _____	
Overall: User Satisfaction (1.0 low - 5.0 high)? _____	

These are pages two and three of the stage end report:

Stage End Report (Page 2)					
Resources & Duration Summary (Express Changes as +/-)	**IT TEAM**	**IT OTHER**	**USERS**	**START DATE**	**END DATE**

Stage Agreement (Original Plan)	**Hrs £**					
Sponsor Approved Plan Changes	**Hrs £**					
Latest Plan (Sum of Above)	**Hrs £**					
Actual	**Hrs £**					

TOOLS AND TECHNIQUES (eg SIMULATION, PROTOTYPING)

Rate from 1 - Very Low To 5 - Very High **Description of Tool/Technique**	**Level of Usage During this Project**	**Usefulness During this Project**	**Familiarity With Tool/ Technique at start**

Expand upon points made above where appropriate and note any other factors that had a good/bad effect on this project below:

On this third page one might record things like this:

- Positive: The sponsor wandered around every couple of weeks asking how it was going and this helped to spur the team's efforts.
- Negative: The users were 100 miles away from the IT people. Next time we should move the IT team to the users' location during the requirements and UFD stages. If we had done this we would probably have finished UFD a month earlier: definitely worth the hotel bills!

And if the project tracked and recorded hours as we described in the planning chapter, attach that gold-dust data to the stage end report. We then have a neat record of what was learned and what could be done better next time, and also a record of where every man hour was actually spent.

This is potentially a very useful document to anyone about to start a similar project and it should be available to anyone who wants to see it. However, with the best will in the world, documents like this do sometimes get filed away on the intranet somewhere and forgotton. Which is why, in companies with a project support person or group, there might be another rule: you must send a copy of your stage end report to project support.

They could simply act as librarian, but maybe we should charge them with a rather more proactive duty: to read each and every stage end report and where a lesson has been learned by project manager A that is clearly relevant to the project that project manager B is about to start up, it is the job of project support to get A and B

talking to each other - or at least send B a copy of the stage end report. And if project manager A's experience would be of interest to the wider population of project managers and team leaders, project support twist A's arm to make a presentation at the next project managers forum. Project support become agents forcing cross-fertilisation.

So, all of the above would be done at the end of each project stage: team post mortem (lessons learned meeting), produce a list of actions you *will* take to make the next stage/project even better, share experience, summarise lessons learned and costs in a stage end report and send it to project support.

At the end of the project

Finally you reach the end of the project. Or more precisely, you reach the planned go live date - the project may not end there as we shall see.

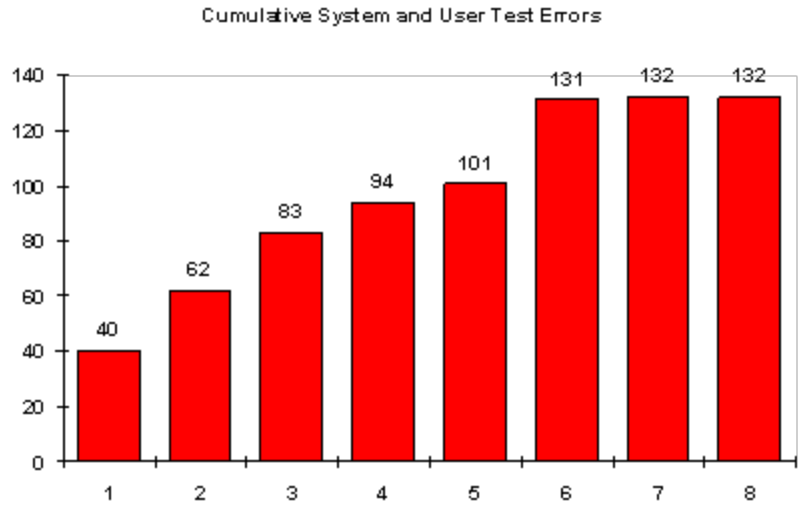
Before the new system goes live there should be a mandatory presentation to the project sponsor, at the end of which the sponsor has to make a decision: whether it can go live or not. This should be done for both in-house and outsourced projects.

Think for a second - if you were the sponsor and your part of the company's business was due to be run by this new system as of Monday morning what would you want the project manager to show you before you'd give approval for it to go live?

Perhaps these sorts of things:

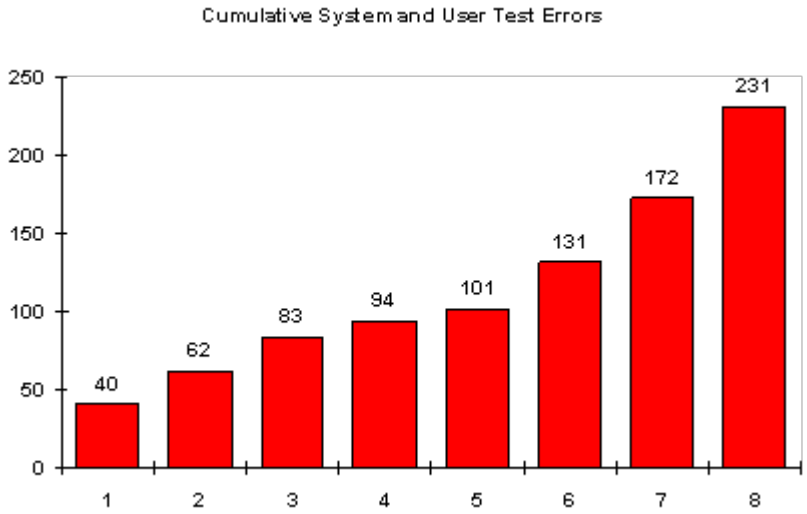
- evidence everything promised has been delivered
- evidence all required sign offs (as specified in the PDD and/or most recent stage agreement) have been achieved, e.g.
    - users
    - IT operations: they're happy to install and support it
    - help desk manager
    - audit
- users have been trained
- a fallback plan exists in case the system doesn't work
- disaster recovery is in place
- evidence the project manager has spent the change budget wisely and hasn't wasted it on 'flashing on and off pink' type change requests
- evidence that the system and other deliverables are fit for purpose
    - evidence that quality has been managed throughout the project
    - evidence that testing is complete as opposed to the end date has arrived
- a prediction of how many (unknown) errors might be in the system which will be found during live running - plus that number multiplied by the typical cost of fixing an error in live running

Let us consider the penultimate item on the list. How would we persuade the sponsor that testing had actually finished? Here is an example of the sort of chart we might show the sponsor:



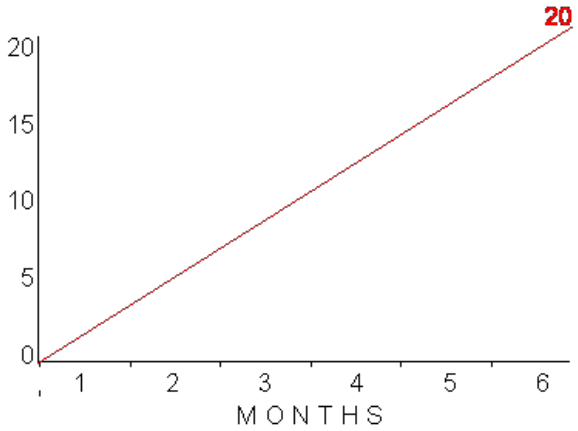
Testing took 8 weeks and a total of 132 errors were found. The message of the chart is that no errors were found during the last two weeks of testing (and we would need to present data showing that testing was still being done in weeks 7 and 8).

If by contrast the chart looked as follows, have we finished testing?



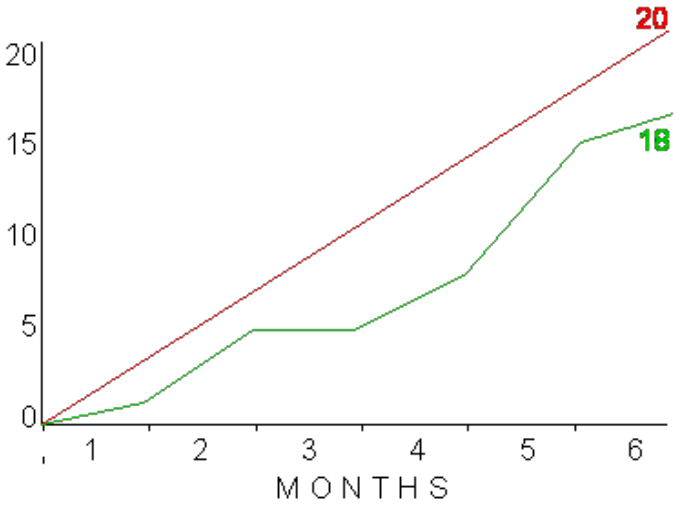
Clearly not.

If you had to predict how many as yet unknown errors might emerge over the first few months of live running, how would you do it? At the simplest you could look back at past system and user tests and see what percentage of the bugs they usually find. So say for example it's normally around 90% and you found 180 bugs in testing this time, you might expect there to be around 20 in live running and you would bring a chart like this to show the sponsor:



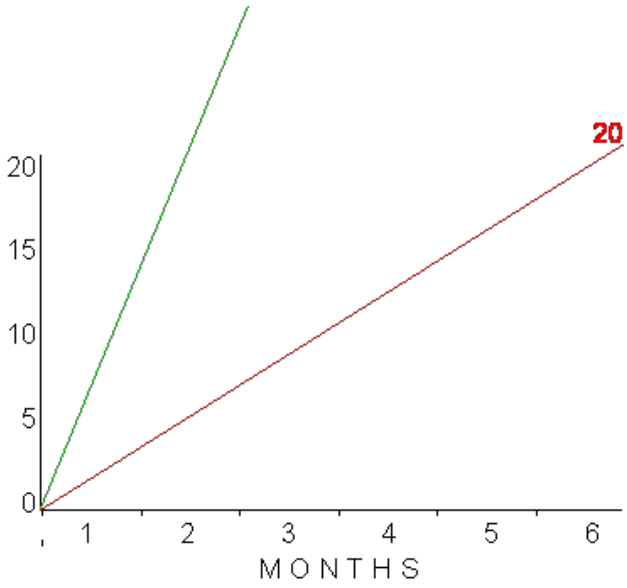
Bugs @£10K predicted in live running

And the sponsor might invite you to put that chart up on your office wall and every now and again he would saunter past looking for the green line, the actual number of errors found in live running. If the chart looked like this after six months the sponsor would probably be quite happy:



Errors predicted vs actual in live running

But if you were the sponsor and you saw the chart at the end of month 2 and it looked as follows, what would you conclude?



Errors predicted vs actual in live running

You were conned at the go live approval meeting.

The appraisal of the project manager should remain open for a couple of months post cutover so that he knows quality will at least be an equal partner with cost and dates when determining his next pay rise. If right from the beginning of the project the project manager knows he will have to predict the number of live errors at the go live approval meeting he has an incentive to manage quality properly so that a) he be able accurately to predict the number: his appraisal depends upon both of these things.

Occasionally as a project's end date nears it becomes apparent that the quality won't be right by the planned end date - there isn't enough time left to get the bugs out. An emotional debate about quality can ensue.

With good quality management something a little more rational becomes possible. "Sponsor, sorry, but if we go live on the planned date of June 1st there will be 100 or more errors in live running and it could cost us £1M to sort out the mess. If we delay by 6 weeks to mid July it'll cost £100K in extra test costs but there should then only be 15-20 errors in live running. What would you like us to do?"

The sponsor might say: "Go live on June 1st as planned. We have a one off opportunity in June to win £500M of new business - we'll bear the £1M failure cost." Or he might say: "Delay till mid-July. The last thing we need is for this thing not to work properly and get us a bad reputation in the market place - that could cost us dear. Get it working properly asap then come back and see me."

The debate becomes more business-like. In extreme cases going live with systems that don't work can drive a company into bankruptcy.

When does the project end? In the PDD or final stage agreement be clear about this. You might say: 'the project ends on the day of cutover.' Or you might say: 'the project ends 2 weeks after cutover' which implies the project team stay together for a couple of weeks to sort out any production problems.

Final stage end report

We mentioned stage end reports earlier. At the end of the last stage the final stage end report should look back over the whole project and record total project metrics as on this template:

Stage End Report (Page 4)			
Use this form with project's FINAL SER to summarise TOTAL project effort. For all stages, attach Effort Distribution Report.			
DETAILED RESOURCE ACTUALS (In Man Hours)	**IT TEAM**	**IT OTHER**	**USERS**
Project Control			
Inspections/M_Checks/Checks			
External Reviews			
Orientation			
Base System Installation			
Define Requirements			
User Functions Design			
IT Technical Design			
System Test Preparation			
Program Development/Unit Test			
Procedure Devel & Unit Test			
System Test			
System Documentation			
User Acceptance Test			
Education Given			

Implementation / Cutover			
Technical Environment			
Risk Adjustment			
Re-work (Inspection/Testing)			
Other:			

It then becomes relatively easy for project support to distil out some of the rules of thumb we saw in the estimating chapter such as the percentage of project effort that is spent in the requirement, UFD, IT technical design, build, test and UAT steps and the relationship between IT effort and user effort in each of those steps.

And project support can do simple things like emailing project managers every now and again saying things like: "you might like to know that our best quality projects invest around 12% of the hours in the UFD step in inspections and simulations. Also, rework following UFD inspections is usually around 9% of the time it took to produce the work in the first place." It is surprising how quickly the knowledge level rises and people begin to assume all humans are programmed at birth with the knowledge that post-inspection rework is around 9% (surely *everybody* knows that...?). In the buzz phrase we become a 'learning organisation'.

User satisfaction survey

Maybe some time after the end of the project users should be surveyed to see how they feel about the new or enhanced system. The survey might ask questions such as:

- Does the system meet the business need?
- Were you or your representative adequately involved in defining the business requirements?
- Were you or your representate adequately involved in reviewing and approving the system design?
- Is the system easy to use?
- Is help and support available when you need it?

And no doubt you can think of a host of other thinks we should ask the users, including 'how could the system be improved?'

Questions such as 'Were you or your representate adequately involved in reviewing and approving the system design?' may prompt the reaction: 'gosh, should we have been involved in reviewing the deisgn? That would have been a good idea!' which could lead to better user involvement in future projects.

If the project manager knows right at the very beginning of the project that this survey will be done at the end, and the results will feature in his appraisal, he'll probably look at those questions at the outset and try and ensure the project does

the things that will yield positive responses to the survey - like involving the users properly at the appropriate time.

At the end of the benefit assessment period

Probably many months after the project team has long since disbanded an independent team look back, not at how well the project was done, but at how good the original business case was by conducting a post implementation review.

Post implementation review (PIR)

Since the project team is long gone the post implementation review (PIR) of the business case might, for example, be run by Finance. Essentially the PIR tries to quantify the actual benefits of the thing the project delivered and assess its actual costs to establish if we're getting the expected return on investment.

Benefits can be easy to measure: if the project developed a new product how many have we sold? The product might be a physical thing or it might be something like a new insurance product where the IT system and the product are more or less one and the same thing and we can easily count how many people bought our new holiday insurance package on the web.

Sometimes it's much harder. If the project's benefit was a percentage saving in manual workload that may not be so easy to measure, but we need to make an assessment.

Occasionally benefits realisation is a major undertaking: the project may have delivered a new system that enables a company reorganisation and rationalisation, but that itself needs to be managed as a project in its own right - two projects delivering one ultimate benefit.

So, under the leadership of Finance, let's say, a team of business users count sales or measure admin workload savings or look at the headcount savings actually achieved after reorganisation and IT look at running costs, bug fixing costs and any other costs associated with operating the system. It's just like building the business case, but after the event. Having done that we compare actual benefits and costs to those foreseen in the original business case created at the start of the project. One advantage of having Finance run the PIRs is they can ensure half a dozen projects don't all claim the same benefit as theirs.

It may nominally be the (ex) sponsor's responsibility to commission this post implementation review but why might he be the very last person who wants it done? It may reveal that the benefits predicted were illusory and the original project cost estimate was hopelessly optimistic. In the worst case, what looked like a good investment for the company at the outset has actually lost us money. We do not want to do that too often.

One of the conclusions of the post implementation business case review might be that we as a company need to strengthen our business case review procedures at the start of projects to be more honest about project costs and more realistic about

benefits so that the Board can make better investment decisions in the future. Indeed, it may well be that it is the Finance Director who mandates that PIRs are carried out for all large projects.

Although the PIR is primarily a financial review, we may also want to assess how well the system (or whatever the project's product was) is performing. Strictly speaking we are not examining how well the project was done, but if it's clear the system is not properly meeting the business needs and requirements, there may be lessons there for how future projects might better understand business needs before charging off and buying packages or writing new systems and hoping they'll be fit for purpose.

Rewards

But let us go back a step. What is by far the most important thing to do at the end of any project? Have a party. Don't forget to invite those who were involved during the early stages who've since left the project team. Most important of all make sure the sponsor is there - with his wallet. Seeing the sponsor handing £20 notes over the bar to buy the drinks is sure to bring a smile to the team's faces.

On the more serious side the project manager has a duty to feed back to the team members' line managers how their people performed on the project. Make sure people who did a good job for you get rewarded by their boss. This, apart from anything else, is in your self interest - they'll want to work for you again.

In truly project-oriented companies there will be significant financial rewards for project managers and project team members who bring large projects in on time, on budget, meet business needs and deliver good quality. Projects can entail hard work, long hours, great stress and the sort of proactive, 'making things happen' approach that may not always be needed when running day to day operations. Process-based senior managers may not always appreciate the sheer energy that goes into setting up and running projects - particularly the energy that goes into those that as a result are made to look easy.

Summary

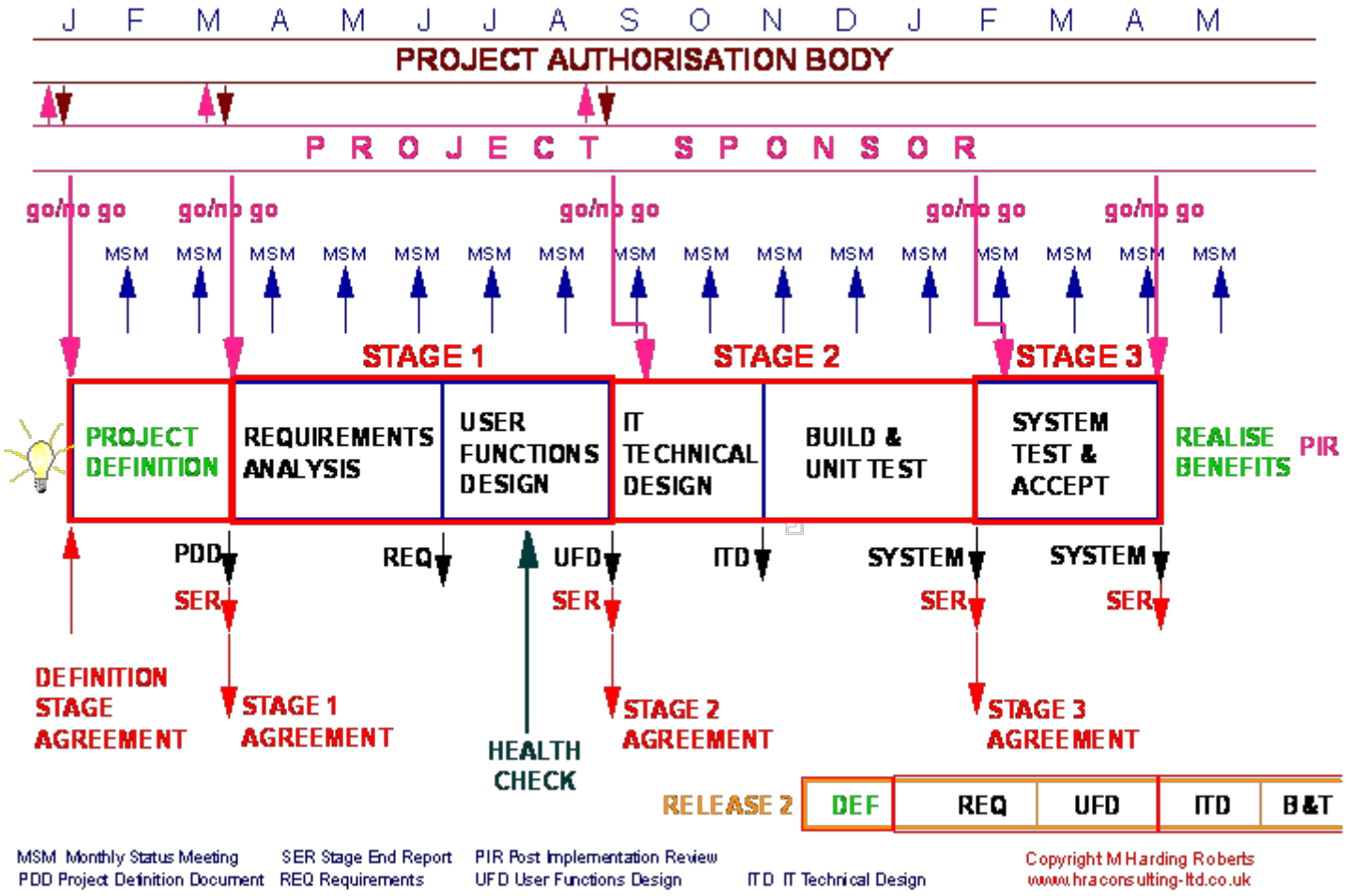
In summary many of these techniques force people to crystalise their experience and learn lessons so that future projects can be even better. Things like stage end reports also enable that experience to be shared with others.

We are trying to find out what it is that prevents all projects being perfect and, one by one, address these inhibitors so that projects get better over time.

Chapter 14 - Project Management Routemap

We have covered a lot of ground - to some readers familiar territory, to others much will have been new. Particularly for the latter, let us see how some of the major themes we have covered might come together and wrap around a project.

Project Management Routemap



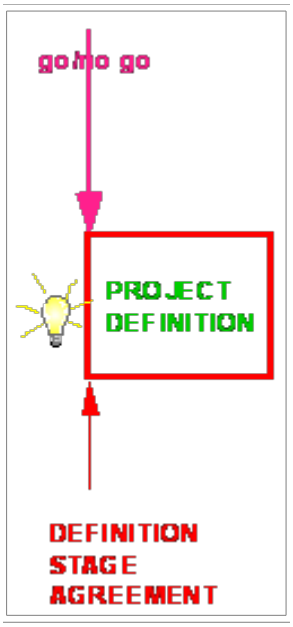
One morning you're driving in to work and you have a mega-brilliant idea: you envision a new IT system that would make your company a fortune. What would you do when you got into the office? (No, not resign, it wasn't that good an idea.) You discuss the idea with a few people and everyone agrees it is a great idea.

And there it might end were it not for the fact that you are a natural, entrepreneurial project manager.

You approach a senior manager who you think would make an ideal project sponsor. The first question he asks you is "what would such a project cost?" But you're not falling for that. You tell him you've got no idea. "OK," he says, "how much would it cost to find out what it would cost?" You're ready for that one. You tell him it will cost £60K to do a proper project definition. Which implies, does it not, that in preparation for your pitch you did a fair bit of preliminary planning for the project definition stage.

The senior manager can't authorise £60K so he takes his great idea (yes, his great idea!) to whoever does authorise project expenditure - maybe the Board of Directors - and they approve a speculative investment of £60K to research and define this potential money-spinning project. Our senior manager has now acquired the mantle of sponsor.

Meanwhile, in anticipation of Board approval, you have been rushing around finalising the project definition plan.



Project definition is going to take 10 weeks. Twenty five people will need to be involved and they are owned by 10 different managers. This is not 25 people full time for 10 weeks. Maybe there will be 2 or 3 full time people to drive it through, business experts spending a few days defining high level needs, IT guys assessing options and feasibility and doing some estimating and Finance guys helping you build the financial justification, etc.

You plan it all out, twist managers' arms and get them to agree they will make people available. You write a stage agreement for the project definition stage saying what you're going to deliver (PDD, business case, project plan, etc) and documenting which managers will provide what resources, and you get those managers to sign it.

You invite the sponsor to pop down and have a chat with your project definition team. The sponsor tells them that his(!) vision is in their hands, that he thinks the project could have big benefits for the company and asks to be kept informed.

Work starts in earnest to give the vision substance, to define the (potential) project. First, was the vision actually just an hallucination? Setting enthusiasm aside, the team step back and ask: is there *really* an opportunity here, was it really such a good idea? Depending upon the circumstances, this could entail market research, competitive analysis, assessing workload savings or any number of other ways of establishing that the need/opportunity is real.

But the sober reflection only strengthens the realisation that this is a very good idea indeed - there is an opportunity for the company to make a lot of money.

Having established the opportunity exists, you set aside the eureka solution you imagined in your car that morning and you consider how else the need might be met. Buy a package? Modify an existing IT system? Develop a new system? Adopt a manual approach? Outsource? Having brainstormed loads of whacky and some sensible solutions, you conclude the best bet would be to write a new IT system from the ground up.

You now start to consult potential users (key ones' time having been committed via the stage agreement). They all think it's a terrific idea and Marketing say "wow, and could it do this sort of thing as well..?" and HR say: "ah, and if it could do this... we could use it for..." and Finance say "and could it also...". Everyone you speak to has something they would like included in this new system. You realise you have a tiger

by the tail and it dawns on you that to try and do everything everyone would like in one go would result in a never-ending big bang disaster project.

So what do you do? You break it into releases and have those tough negotiations: "sorry, Marketing, your stuff will be in Release 3 (maybe)". The scope of Release 1 thus begins to emerge from the mist. Perhaps you now do some high level process modelling to try and get a clearer view of the things you are going to include in Release 1. That helps you understand what it will take fully to analyse the detailed requirements for those things so that you can start to do some estimating and planning for the Requirements Analysis step.

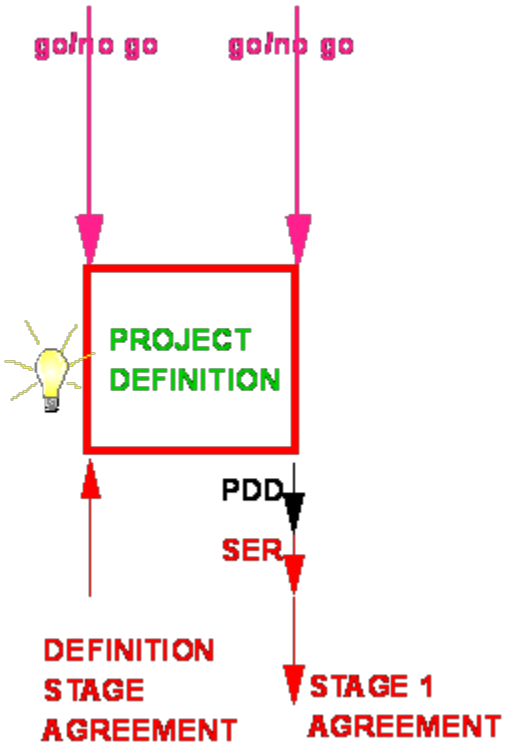
If you're smart (let's assume you are) one of your project definition team will be your project leader and he has time in the project definition plan to help you do the planning of the project proper. The scope of Release 1 is now fairly well pinned down so you and your project leader rough out the high level plan for Release 1 on the back of an envelope and reckon three management stages would be appropriate: Stage 1 Requirements and UFD, Stage 2 IT Technical Design, Build and Unit Test and Stage 3 System and User Test and Implementation. The IT guys come up with some estimates, you use some rules of thumb to work out what the user costs might therefore be, think about the risks and 'unexpected' costs you might end up incurring in the project then get the whole estimate looked over by project support and the man hours translated into money by Finance and conclude Release 1 will cost around £1M - including bags of contingency given the number of unknowns there are at this stage. The users on your project definition team estimate the bottom line benefit that will accrue to the company from the new system and think it will be at least £1M a year. A pay back within 12 months - pretty good.

You go and see the sponsor and tell him what you think Release 1 will cost. He says: "*How* much?" but he approaches the Board who approve in principle a £1M budget for Release 1.

It starts to become clear who you will need on the project steering committee, who would make a good project user manager, etc, so you sit down with those people and describe the roles you'd like them to fulfil, get their backing for the project, agreement to scope, etc. You and the project leader flesh out the plan for Stage 1 (Requirements and UFD) and start to get outline resource agreements with relevant line managers. You set the Finance guys to work building the formal project business case.

You are now in week 8 of the 10 week project definition stage. You hold a (pre-planned) project definition workshop (PDW). Sponsor, steering committee members and project role players all attend.

You run through scope, roles, plans, estimates, resource requirements, risks, how you'll manage issues and changes, what you'll communicate to whom, what might be in Releases 2 and 3 and so on, and due to your assiduous lobbying prior to the meeting everyone present agrees to everything (if only it were so - in practice there will be issues to resolve despite all the lobbying).



You, or more likely one of your project definition team, write the project definition document (PDD) to record what everyone agreed at the PDW.

At the PDW the Marketing Director (one of the steering committee members) said: "sure you can have 3 guys for 6 weeks to define detailed requirements". Fine. But you must now finalise agreements with their immediate managers which 3 people and precisely when they will be available. You will have spoken provisionally to the resource owning managers earlier so this should be a matter of firming up the outline agreement you've already got.

At the beginning of week 9 you hire your quality leader and set him to work with the project leader to make sure quality assurance tasks are solidly included in the plan. The detailed task by task plan for Stage 1 begins to crystallise - you can now estimate Stage 1 bottom up and by hook or by crook get a clearer top down estimate for the remainder of Release 1. You add it all up and hope the answer is still around £1M (it is - but if it wasn't, you'd need to tweak the business case and have a quiet word with the sponsor). The more you know about the project the higher the estimate will be - which is why any initial guesstimates must have large amounts of contingency in them to allow for things you couldn't specifically itemise that early on.

So where are we? You've just about got everything done. You've agreed scope, roles and resources. You have a detailed plan for Stage 1 and outline plans for Stages 2 and 3. The Finance guys have helped build the business case and perhaps prepare a month by month project budget. You've agreed which people will be available and when to work of on Stage 1: users, IT and anyone else we'll need.

You now write the stage agreement for Stage 1: what Stage 1 will deliver and when (requirements document, UFD, Stage 2 plan, revised business case, testing strategy, etc); risks; milestones; and most important of all: resources - who when; and maybe things people have agreed to deliver to you: before the end of Stage 1 you'll need the technical development environment set up and ready to go. You get resource owners to sign the Stage 1 agreement.

In reality these last 2 weeks of project definition will entail an awful lot of problem solving, negotiation and issue resolution. There will also be a lot of setting up to do: office space, time recording mechanisms, change control procedures, etc. The setting up of all of these will of course have been tasks in your project definition plan.

In week 10 you get the team together for a post mortem. How did we do? Could we do future project definitions better? The team distil out lessons learned and capture them in the stage end report (SER). If you have been time recording during the

project definition stage, you attach a record of where the hours went. You might even attach a bar chart showing the plan you would have produced for the project definition stage with the benefit of hindsight.

When planning a project it's quite useful to start at the end and work backwards. At the very end you'll need a user satisfaction survey - when should that be produced? You'll need a handover-to-production checklist - when should that be produced? You'll need test data and before that test scripts and before that a test plan and before that a testing strategy - when will each need to be produced? All of these things - and many, many more - will feature in your outline plan for Stages 2 and 3 of Release 1 and help you work out costs and when you will need people to join the team. Throughout Stage 1 keep revising this map of Stages 2 and 3 as new things occur to you (they will).

At the end of week 10 you have the summit signing ceremony - the formal go no go - with the sponsor and anyone else he chooses to invite, perhaps the Finance Director and IT Director.

Your presentation demonstrates the project is indeed well defined and everyone has agreed the project definition. If you have kept the sponsor in the loop it should be a matter of saying: "Sponsor, as you know, scope has been agreed, Finance have signed off the business case..." etc. But there is one bit of bad news you must tell the sponsor about. Having added up the hours everyone spent on the project definition the actual cost was £80K versus your budget of £60K. The sponsor wraps your knuckles, tells you not to do that again, but nevertheless says you've done an excellent job turning his vision into a solid project.

You then present your Release 1 plan: a summary of the detailed plan for Stage 1 and your outline plan for Stages 2 and 3. You estimate Stage 1 will cost £400K and Stage 2 and 3 very roughly £600K - but you stress this is not a commitment, just your best estimate on what is known at the moment. To demonstrate that you have resource commitments for Stage 1 you then call in the waiting line managers and at your behest the sponsor asks each in turn if they really can make available the bodies shown on the Stage 1 plan, and they all commit eyeball to eyeball to the sponsor and then sign the Stage 1 agreement. Back out of *that* commitment if you dare.

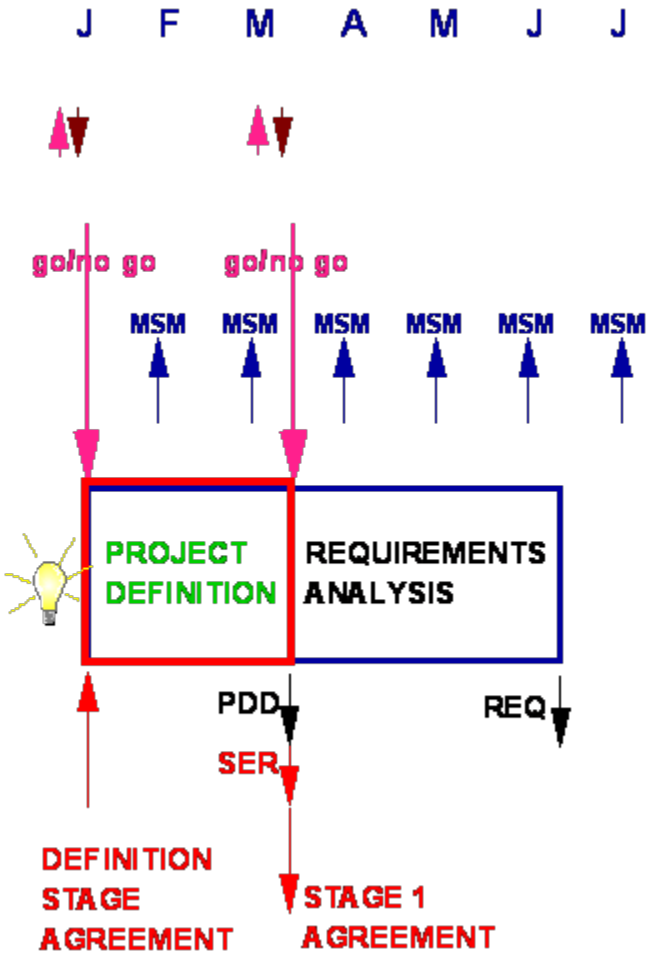
The sponsor authorises £400K for Stage 1 and says: "As soon as you think it might cost outside a range of £350K to £450K come and tell me. Don't wait till you have overspent - is that clear?" Yes, sir.

And he gives you a cheque for £1000 - not for you personally but to take the team out to celebrate a project definition well done.

Did we take a risk by investing in all that project planning and for example in hiring the quality leader before we got the go ahead? Not really. In practice it becomes evident long before the end of project definition whether the project is going to fly or not. If, say, in week 6 it became clear the project was infeasibility, or would have a negative return on investment, or the company didn't have the people to do the project, you'd wrap it up then, hold a post mortem, produce a SER and get back in your car in search of a better brainwave.

You can perhaps see why a project definition stage for a large project needs careful planning. A lot of people have to do a lot of disparate things in the right order in a limited amount of time to get a project well defined. If project definition isn't carefully planned it will be chaotic and, more important, the project will not get properly defined or planned - and it's failing before it's even started.

But our project is well defined.



After the meeting the sponsor calls you back for a private discussion. He tells you that if you bring the project in on time, on or around budget and it works, he has agreed with the Board that you will be paid a large bonus and that another amount of money will be made available for you to disburse as bonuses to the project team at your discretion.

The project proper begins. You organise a kick off meeting. The sponsor tells the team - IT people and users together - the project is important and they, the team, hold a key part of the company's future in their hands and he wants it done on time, on budget and top notch quality. And if anything or anyone gets in the way of the project, just let him know. The team are enthused.

Ideally core members of the project team will have worked on the project definition so they have a good understanding of what's really needed from the project. Also, ideally, they will become the team leaders during the build and test stage and so carry that understanding right through the project.

The tough job of engineering or re-engineering business processes begins. Intensive workshops force the business users to define every item of data the system will have to handle (length, valid values, etc) and every operation to be carried out on these data (lookups, calculations,...), including exception routines, special cases, what happens when things go wrong, etc.

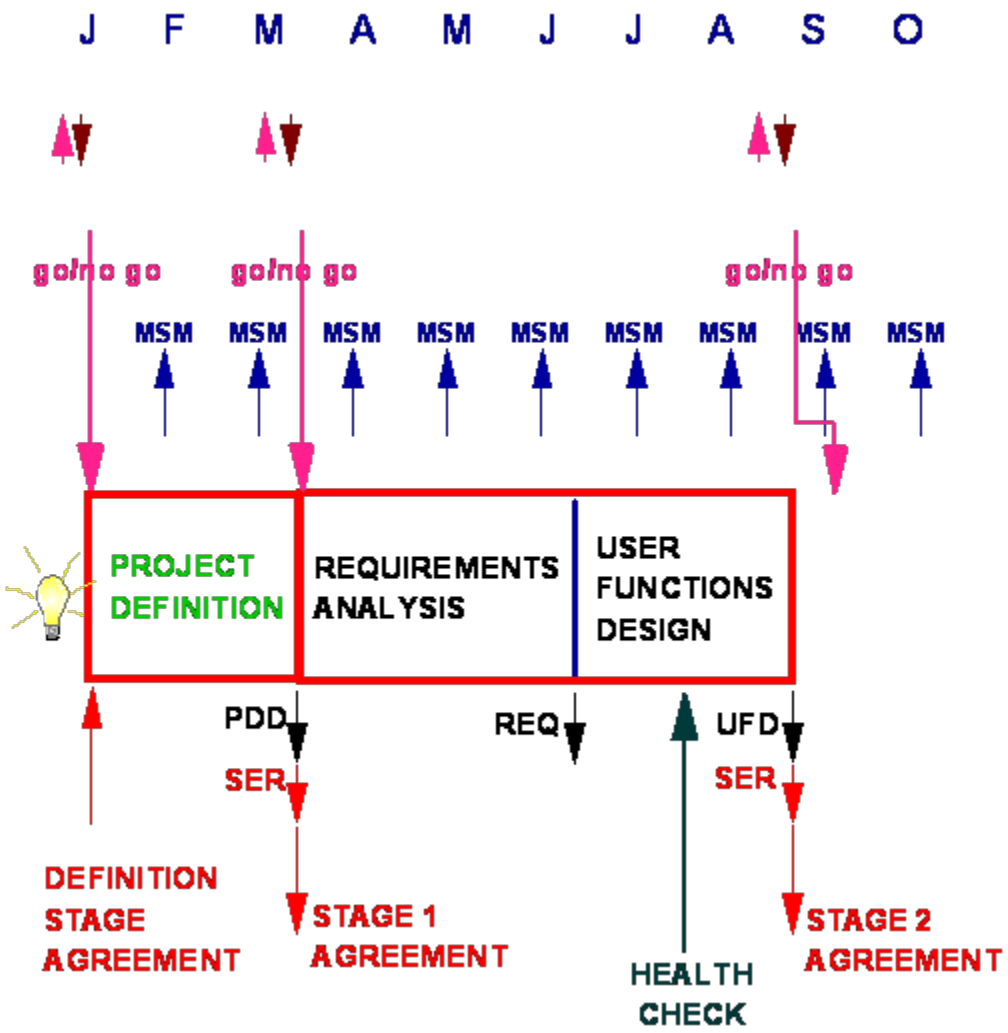
Luckily you have a tough business analyst running these sessions who calls a timeout whenever a user is unable to say exactly what is required and makes them go away and find out. This encourages them to come better prepared next time.

By the end of the requirements analysis step you must believe you won't need to ask the users anything else about what they want. Don't be tempted to leave tricky requirements to be defined later. To do this is to litter your future path with bear traps of your own making.

Every month you report status to the project sponsor in a monthly status meeting.

The team conduct thorough inspections of each section of the requirements document as it is finished. When the whole document is completed they run a simulation - pumping real test data through the business process, performing the validations and calculations, writing down the data that will appear on the outputs of the business process. Many problems are found. Most are gaps - the business process doesn't say what should happen in particular circumstances. It takes two weeks of stopping, plugging gaps and restarting to simulate the whole business process from end to end. Naturally this was in the plan.

The requirements document is published. The Stage 1 agreement will have said who will sign it off and how long they'll take (2 weeks?) to do it. Ones hopes they will have been involved all the way through the requirements analysis step and they or their people will have been involved in inspections and simulations so they *know* the requirements document is OK before they even get it.



Work begins on the user functions design (a.k.a. functional spec, external design,...). Joint user/IT teams evolve the user-visible functions of the system: screen/web page layouts - should the 15 data items that comprise an order be input on one screen or several? Output document layouts - how should the 21 data items that comprise an invoice be laid out? The IT team design all the invisible user functions and the system's infrastructure.

At some point during the UFD step the techies, who will later do the IT technical design, do a quick preliminary technical design just to make sure what is being proposed in the UFD can be built. Indeed they may suggest changes to the UFD to make it simpler/easier/cheaper to build.

As each UFD section is completed it is inspected - users, other UFD designers, technical designers and test team members are all involved.

Once the UFD is finished, a major simulation exercise is undertaken. Someone pretends to be a customer accessing the order entry web page. Someone else writes down the data that the 'customer' entered (e.g. customer number 326-086). The

simulation team perform the edits specified in the UFD. Data is stored on flip charts. Output documents are compiled and written out by hand. This is simply people pretending to be a computer, simulating what the machine will do when the code is written. Areas that caused concern when the business processes were simulated during the requirements analysis step are given special focus.

If you can do the simulation you probably have a reasonably good UFD. If you can't do a simulation because the UFD is vague, incomplete, wrong or incomprehensible, *stop the clock*, delay the end date of the UFD step and get the UFD right. If you carry on regardless you'll just have a much bigger delay later.

The quality leader is strong and ensures all the planned quality checks are conscientiously performed.

Midway through the UFD step an independent team come in and do a health check. They conclude you're pretty much on schedule, on budget and the evidence suggests the UFD will be a high quality document: complete and correct. The health check team give your project an A rating.

Six weeks or so before the end of Stage 1 what must you do? Start planning Stage 2 in detail, negotiating for resources, assessing risks, etc. You estimate the cost of Stage 2 bottom up and revise the top down estimate for Stage 3. You add those two numbers to what will have been spent on Stage 1 to get a revised total project cost.

And does it add up to £1M? Er, no. It comes to £1.3M. Do you wait until the formal go no go meeting at the end of Stage 1 to spring the surprise on the sponsor? Definitely not. You go and have a word with him now, a few weeks before the end of Stage 1. The extra £0.3M, however, is way beyond even his tolerance so he goes to the Board and manages to acquire the extra £0.3M.

You publish the UFD for sign off - which again should be a formality given the users' involvement in the inspections and the simulation.

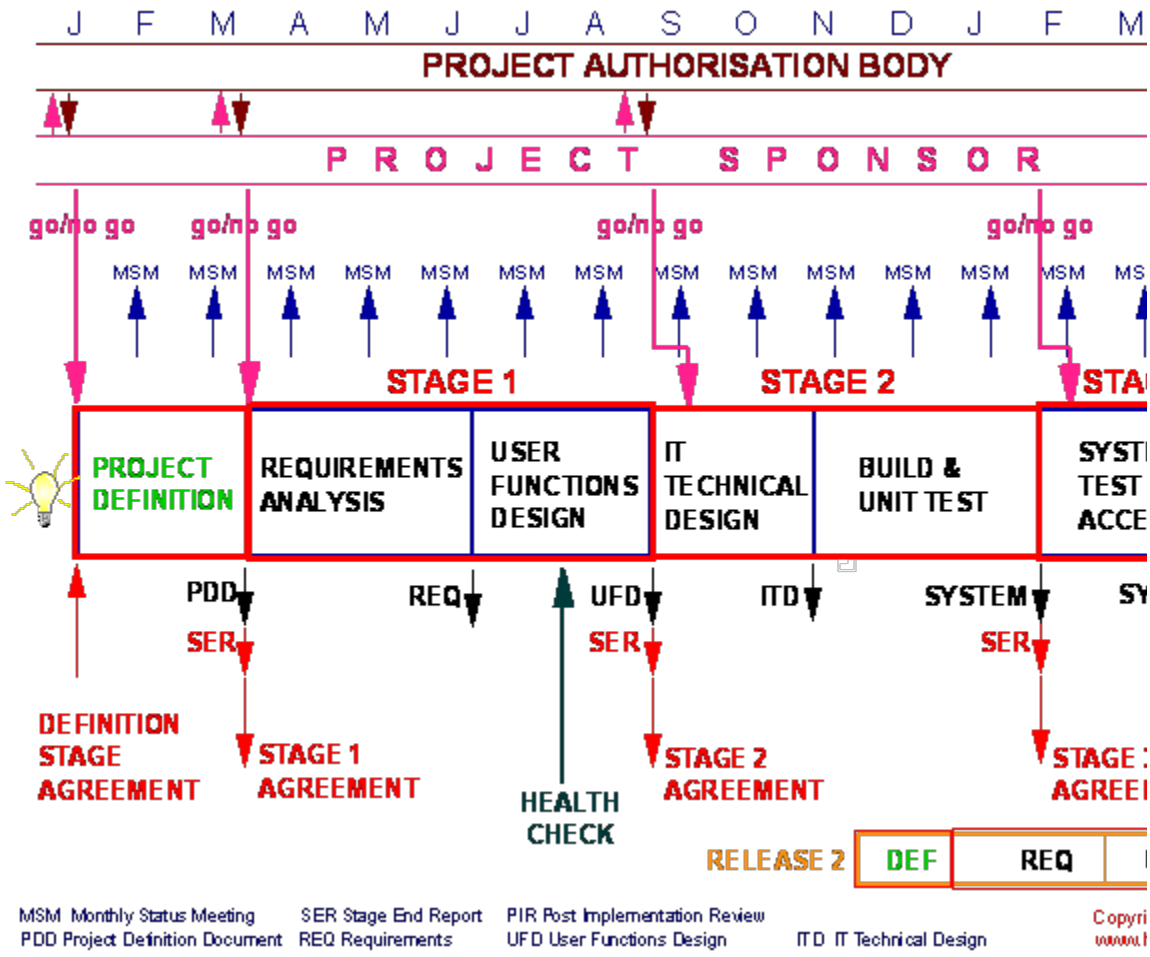
Line managers commit resource via the Stage 2 agreement. You hold a post mortem and produce a stage end report for Stage 1.

Why is there a 2 week delay before the formal go no go meeting at the end of Stage 1? To allow for issue resolution, time to get sign offs, etc. But of course work begins on the IT technical design in the meantime.

At the formal go no go meeting you tell the sponsor you've achieved all the required UFD sign offs and there are no open issues (if only it were so!) and that Stage 1 actually cost £445K - just within tolerance. This figure does not come as a surprise to the sponsor as you will have been revising your prediction of this figure at each monthly status meeting.

You prove you're ready to start Stage 2 and the sponsor approves its budget of £600K +/- 10%. IT technical design and build run like clockwork because the UFD is complete, correct and stable. Integration test goes remarkably smoothly due to the very detailed planning dovetailing build, handover and component testing. In the

penultimate monthly status meeting of Stage 2 you predict Stage 2 will probably come in a little under budget at around £550K.

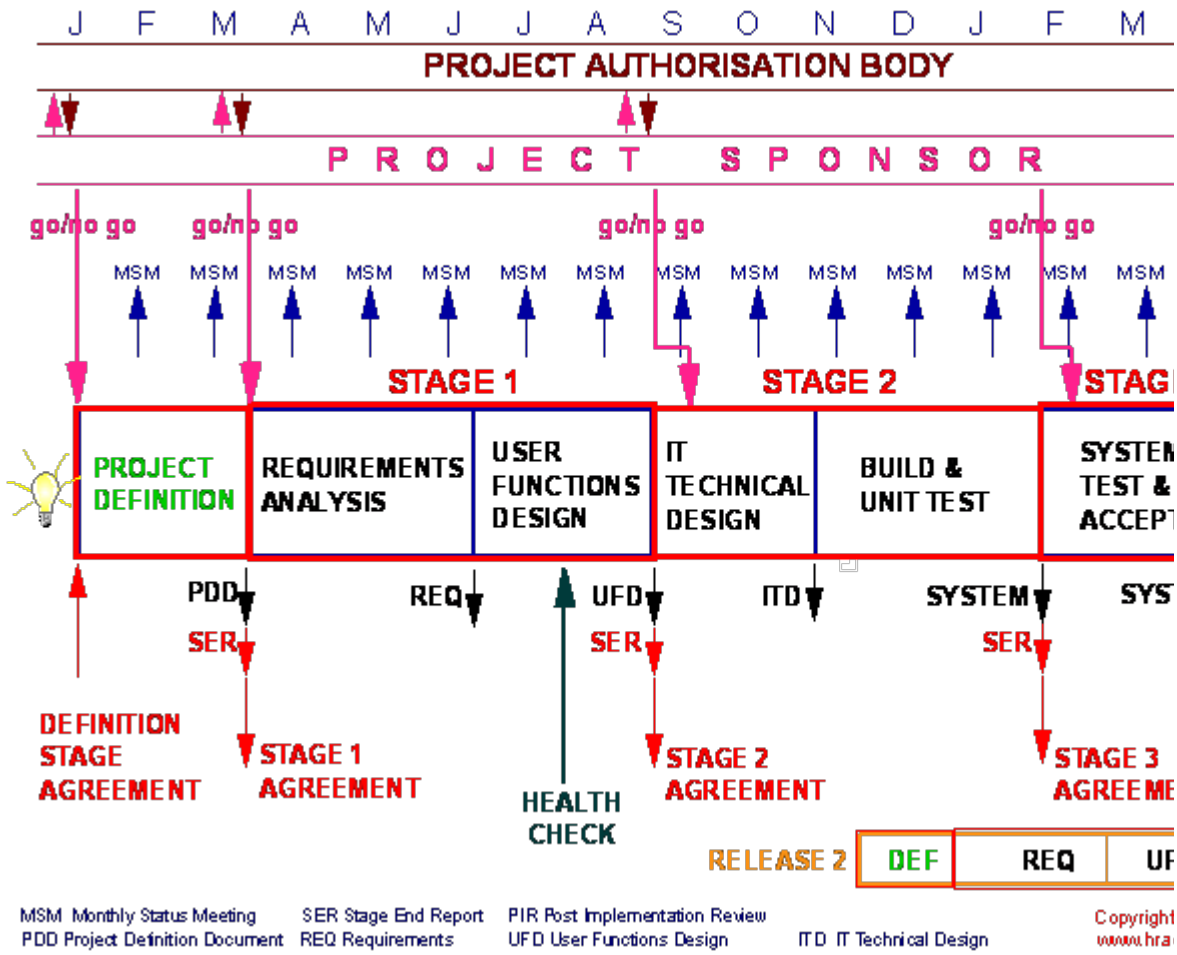


At the end of Stage 2 the pattern is repeated. What is the implication of the fact there are no double arrows up to the 'Project Authorisation Body' at the end of Stage 2? You still think it's going to cost around £1.3M. The sponsor does not need to go back to the Board with his begging bowl. But why might the sponsor cancel the project at this meeting? Because the projected benefits have evaporated - maybe somebody beat you to that market opportunity. This is why we should re-examine both costs *and* benefits at the end of each stage.

But our project's benefits are still there. Having shown you have Stage 3 planned, resources committed, etc you get the go and a Stage 3 budget of £250K.

However, a remarkable thing happens during Stage 3. The system test plan took no account of all that quality checking you invested in up front. System test finishes two weeks early as you have run out of bugs to find. And this is two weeks earlier than *planned* - probably a couple of months earlier than everyone was expecting given the usual slippages that such projects encounter. System tests always overrun, don't they? Not if you really do get the design complete and correct. Users are trained, roll out is readied, system documentation is finalised.

ust before the system is due to go live you present to the sponsor and show you've delivered what you promised, users have been trained, help desks are raring to go - you really are ready to cutover. You show the evidence that testing has finished. You predict around 20 bugs will emerge during the first 3 months of live running and make the point that you would have expected 50 or more bugs if the users had not got involved in the requirements and the UFD inspections and simulations. The final cost is £1.22M, well within the final budget of £1.3M. The sponsor gives permission for the system to go live.



You feed back to the line managers of people who have worked on the project to ensure their project performance is reflected in their next appraisal. At the post project party the sponsor visibly pays for the drinks. A month or so after cutover, when it's clear the system works properly and there aren't too many bugs a large bonus cheque finds its way into you bank account and smaller ones into the bank accounts of team members.

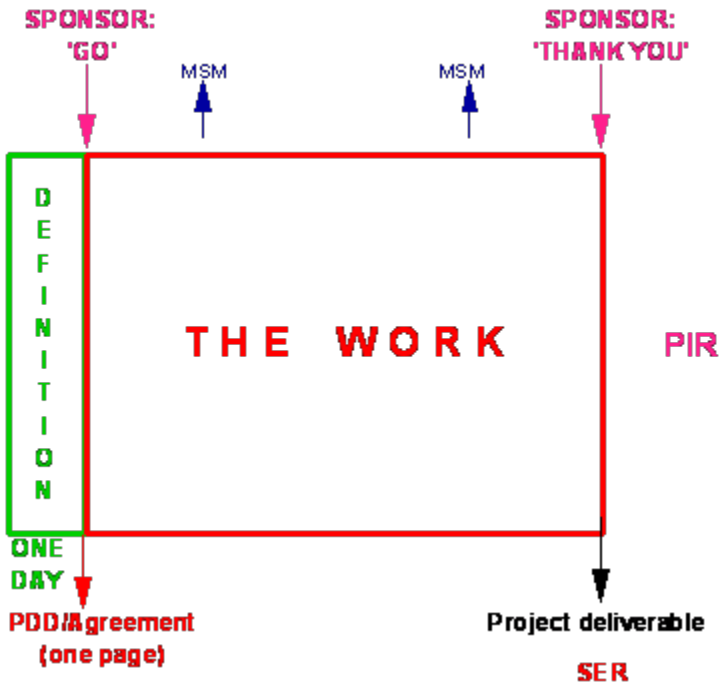
Six months down the line the post implementation review shows that the idea you had in the car that morning was indeed as good as it seemed - the system has resulted in extra business cascading in.

Meanwhile, Release 2 started up under another manager in parallel with Release 1 build. Although the original PDD suggested what might be in Release 2, nine months on the world has changed and the Release 2 project definition could conclude it should contain quite different things. You hope the manager of Release 2 follows your lead and invests in getting it right in the early stages. The danger is he thinks Release 2 will be easy because you made Release 1 look easy, not put in the effort you did, deliver a poor quality Release 2 and undermine the system's reputation and yours. But one hopes Release 2 will be as well managed as Relesae 1 and will be another successful project.

That's how large-ish projects might look, what about smaller ones?

Small projects

If you're lucky your project will look more like the one below. A couple of days is sufficient to define and plan the project, rather than ten week's work. A single side of A4 suffices as the PDD/stage agreement/plan combined. The sponsor says: "Yes, that's exactly what I want you to do." You tell him how it's going every now and again. At the end you say: "there it is, boss." He says: "thank you." And that's it.



Whether it is at one extreme as simple as this or at the other extreme as complicated as the earlier chart, or even much more complicated than that, it is exactly the same in principle. The great skill of the project manager is scaling the controls to the need of the project rather than laying the same bureaucratic structures on all projects whether they need them or not.

The key objective of any projects is to deliver specific outcomes on time and within budget. This apparently simple goal is surprisingly difficult to achieve according to countless research reports. The following 10 questions test your knowledge and wisdom in project management. Answers will be given at the end of the quiz.

1. What most distinguishes projects from day-to-day operation?

- (a) It has a definite budget.
- (b) It has to deliver specific outcomes at a predefined time.
- (c) It brings changes to the organization.
- (d) It requires planning.

2. If you are only allowed one tool to manage a project, which of the following would you choose?

- (a) Work breakdown structure
- (b) Gantt chart
- (c) Resource chart
- (d) Risk register

3. Do you agree project management is an isolated discipline being used by only a few specific industry sectors?

- (a) Yes. Only industry sectors such as construction and IT are using project management.
- (b) Yes. Project management requires lots of mathematical skills that only engineers can handle.
- (c) No. Although not essential, it is trendy to apply some project management concepts.
- (d) No. Project management is now adopted by all industry sectors for projects large and small, and is absolutely essential for successful project delivery.

4. Risk is a difficult subject in project management because:

- (a) Risk involves something unknown.
- (b) People tend to believe a risk will not occur until it actually does.
- (c) Some risks are unpredictable and hence impossible to manage.
- (d) Some risks such as natural disaster or fire are impossible to prevent.

5. Changes are inevitable for every project. The implication is:

- (a) Project planning is not important as it will always be changed later.
- (b) A proper change control process has to be implemented.
- (c) It is mandatory to have customer signoffs for all planning documents to avoid subsequent changes.
- (d) Skills in handling changes are more important than those in project planning for a project manager.

6. The biggest challenge to a project manager in team management is:

- (a) Most project team members are expert in specific areas and consider the project manager not technically competent.
- (b) Many team members are employed on contract basis and busy looking for another job near the end of a project.
- (c) There are too many communication channels both within and outside the team.
- (d) Most of the time a project manager has to manage the team without formal authority.

7. Which of the following factors can definitely make a project fail?

- (a) The project team does not have insufficient skills.
- (b) The customer raises a lot of change requests.
- (c) The project team fails to communicate and build up working relation with the customer.
- (d) A key team member is pulled away from the project due to changing priority.

8. Quality, schedule and cost are three important objectives of any projects. Which

of the following statement is true?

- (a) Their relative importance compared to each other varies from project to project and has to be decided by the stakeholders involved.
- (b) Quality is of equal importance to schedule and cost in today's projects; though in the past schedule and cost take precedence over quality.
- (c) They are usually called the triple constraints of project performance.
- (d) Quality is a performance objective for customer, whereas schedule and cost are performance objectives for the project team.

9. When a project needs to procure equipment or services from the outside, vendor management is a serious challenge for the project team because:

- (a) Vendors are motivated by self interests.
- (b) Contractual matters involve legal knowledge which is a difficult subject for most project teams.
- (c) Vendor usually pursues a different interest than the project team. For example, vendor is more concerned about profitability whereas the project team is concerned about on time delivery of the project.
- (d) Vendors present a security threat to the project team, as proprietary company information may easily be leaked to the outside world.

10. Which of the following statements about 'power and politics' is true for projects?

- (a) Power and politics are unavoidable in any projects because stakeholders?interest and sometimes power will be affected. The only way to balance interests and power among stakeholders is through political process.
- (b) Power and politics are to be avoided in order to deliver a successful project.
- (c) Politics are distasteful to most people and damaging to organizations. It is critical to build up stakeholders' relationship to such an extent that no politics exist within a project environment.
- (d) A project team should focus on technical deliverables of the project. As long as people get good results from a project, they will not turn to politics to achieve their needs.

Answers:

1. ***What most distinguishes projects from day-to-day operation?***
while the correct answer is (c)

Operation and projects alike require budget, planning and outcomes. On the other hand, every single project brings about changes to the organization involved.

2. ***If you are only allowed one tool to manage a project, which of the following would you choose?***
the correct answer is (a)

Although Gantt chart, resource chart and risk register are essential to today's project managers, WBS defines the entire scope of a project and is the source from which activities, resource needs and risks are derived from.

3. ***Do you agree project management is an isolated discipline being used by only a few specific industry sectors?***
correct answer is (d)

Project management is used by virtually all companies nowadays. Every organization needs to deliver business results and this can be achieved by beefing up PM skills of managers and adopting PM best practices.

4. ***Risk is a difficult subject in project management because:***
Correct answer is (b).

Risks involve unknown factors, and some difficult to predict and prevent. However they can always be managed. The most detrimental factor in risk management is people's attitude towards risk.

5. ***Changes are inevitable for every project. The implication is:***
the correct answer is (b)

Good project planning and involvement of customers in accepting planning deliverables help reduce changes in the latter stages of a project. However, a good change control process can ensure the project's objectives can be met even though changes inevitably occur.

6. ***The biggest challenge to a project manager in team management is:***
the correct answer is (d)

Whereas the other three answers are common people issues within a project environment, managing without authority (or formal authority) is a real challenge for project managers.

7. ***Which of the following factors can definitely make a project fail?***
while the correct answer is (c)

While the other three factors can affect project performance, poor communication and relationship between the project team and customer will definitely break a project.

8. ***Quality, schedule and cost are three important objectives of any projects. Which of the following statement is true?***
the correct answer is (a)

Quality is now regarded as of equal importance to the traditional triple constraints of a project-scope, schedule and cost. In a real-life project, there is bound to be tradeoff among the four objectives. Their relative importance has to be decided by the relevant project stakeholders.

9. ***When a project needs to procure equipment or services from the outside, vendor management is a serious challenge for the project team because:***
Correct answer is (c) .

The key challenge of vendor management is to align the vendor's interest with the project's objectives. Unfortunately that is not always easy to achieve, as vendors tend to be more concerned about profits and financial performance, whereas project teams focus more on time, scope, and quality.

10. ***Which of the following statements about 'power and politics' is true for projects?***
the correct answer is (a)

No matter how people disdain politics, they exist in any projects regardless of their size. A project manager has to maintain strong politicalties throughout the organization to achieve project success. Use of politics cannot be ignored or avoided.