Spreadsheet Modelling Best Practice

by

Nick Read and Jonathan Batson

BUSINESS DYNAMICS

April 1999

For more information contact Jonathan Batson on +44 (0)20 7968 5398 or Nick Read on +44 (0)20 7968 6629

Table of Contents

1 Introduction

The spreadsheet is an enormously flexible and powerful tool. It is used by almost every organisation and nearly every business decision of any importance is backed up with a spreadsheet model of the financial projections.

But there is a big difference between the best and worst examples of spreadsheet modelling. For example:

- do you ever worry that you do not understand exactly how your spreadsheet model is working?

- do you ever uncover major flaws in a model days before an important decision?

- do you ever end up abandoning a complicated spreadsheet?

- do you ever have doubts about the accuracy of the results of your spreadsheet?

- do you ever think that you are not getting the most out of the time and effort spent building a complex model?

- do you ever wish that your spreadsheet model could answer the questions you really wanted answered?

You are probably right to be concerned. In the 1999 IBM Business Consulting Services survey of spreadsheet modelling, we found that, for major transactions with large sums of money at stake, most modellers have no formal training in good modelling techniques and that their organisations do not have even the most rudimentary internal standards.

What can you do? The answer is a considerable amount. By taking steps to understand and control the process of spreadsheet development, define the objectives and calculations, design the spreadsheet so that it is easy to understand, test the results and use your model effectively you can achieve much more with your spreadsheet.

Spreadsheet Modelling Best Practice is a guide to developing high quality spreadsheets. This guide is of interest to anyone who relies on decisions from spreadsheet models. The techniques described include areas such as ensuring that the objectives of the model are clear, defining the calculations, good design practice, testing and understanding and presenting the results from spreadsheet models.

So, how do you recognise a best practice spreadsheet model?

The benefits of best practice modelling

A best practice model is:

- easy to use, so you can be more productive in using the model for analysis - rather than struggling just to produce simple results from a badly designed model;

- focussed on the important issues, so you do not waste your time in unnecessary development;

- easy to understand, by using a transparent design you can readily understand the effects that characterise the business problem – and understand them better than your competitors; and

- reliable, so that your model becomes the accepted tool for calculating results - increasing your influence in negotiations.

You can achieve these benefits by applying some straightforward recommendations and techniques whenever you use a spreadsheet package.

Spreadsheet software

This methodology is appropriate for all major spreadsheet packages. However, for consistency all of the examples in the guide have been created with Microsoft Excel 97 and use Excel's notation for writing formulae.

If you use another spreadsheet package, such as Lotus 1-2-3, you will find that the precise notation used in the examples will differ slightly from what you are used to. All of the principles described are equally relevant for other major spreadsheet packages.

2 The Modelling Life Cycle

All computer models go through different stages of development, but the accessibility and ease of use of spreadsheet packages sometimes make it easy for these stages to become blurred. Experience of the way that models grow and evolve will help you develop better models.

In this chapter, we:

- introduce the six stages of model development on which modelling best practice is based;
- define some of the different roles that people play in model development;
- consider the risks involved with modelling; and
- discuss different types of modelling projects and how the modelling life cycle differs between them.

Stages of the modelling life cycle

This guide is based around six stages in the life of a model, illustrated in the diagram below.

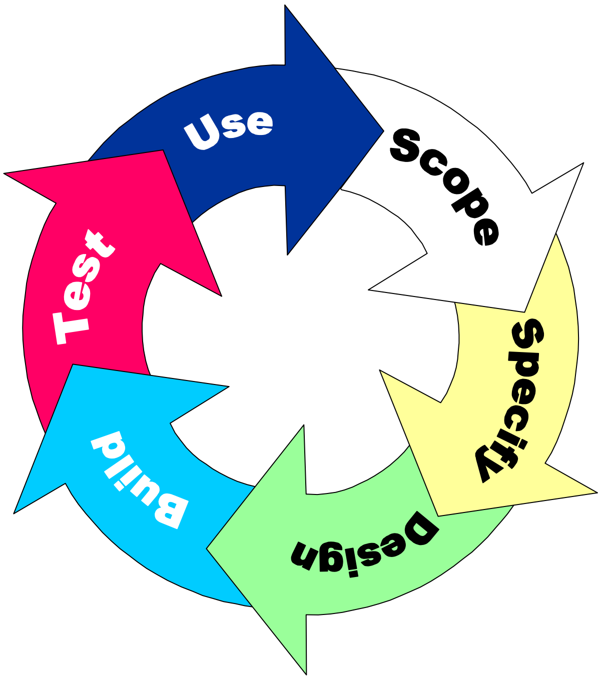


Figure 1: The six stages of the modelling life cycle

Some of these stages are similar to those you might expect to find in a more traditional systems methodology. This is no coincidence, since our best practice guidelines have drawn from other methodologies to produce one tailored to spreadsheets. Where the stages of the modelling life cycle are significantly different is in the flexibility with which they should be used for different types of models. It is this flexibility that is one of the spreadsheet's key strengths, but it is also a risk. Spreadsheets are easy to use, but they are also easy to abuse.

Below we consider each of the six stages in turn.

Scope

This stage is where we assess the nature, scale and complexity of the model required. During the scope stage you should:

- decide what needs to be included in the model and what can be omitted;
- consider the level of detail required in the input and logical assumptions;
- understand in outline how the model will work;
- estimate the time and resource required for the model development; and
- agree the above with the key stakeholders.

Specify

To specify is to define the logic of the model in sufficient detail to provide an unambiguous statement of how the results will be calculated.

For modelling projects where the logic of the model is difficult to define, specification is time consuming and this stage may need to be revisited a number of times.

Design

The design stage involves producing the most effective structure for the model. By following some key guidelines it is possible to make your spreadsheet models significantly easier to use and less prone to errors.

Build

The build stage is where the actual coding of the model takes place. Many spreadsheet models start their life cycle here with only scant regard to the three stages outlined above. This leads to:

- a model that is more complex than originally expected and takes longer to build;
- assumptions being made that were not part of the original brief; and
- a lack of common understanding about what the model is doing.

Test

To test a model is to root out errors and inconsistencies and to increase confidence in the results that the model produces. While it is never possible to test a complex model to the extent that you can guarantee it to be error free, this stage does provide the confidence that the model is producing reliable results.

Use

A best practice spreadsheet model is a powerful tool, but to realise its usefulness you need to understand how to:

- present useful information to help make decisions, not just print out numbers;
- present sensitivities and scenarios to understand the important drivers in the business; and
- control the evolution of the model when further changes are required.

Best practice models

We said in the introduction that best practice models are easy to use, focussed, easy to understand and reliable. How do you achieve these goals by using the above stages in the modelling life cycle?

A best practice model is…

Easy to use	• Using good **design** makes the model easy to operate, while a clear **specification** will explain how the model works. • The techniques in the model **use** chapter will let you get even more from your model.
Focussed on the important issues	• Spending time on the model **scope** will make sure that your model answers the right questions.
Easy to understand	• Good **design** and **build** techniques make your model easier to understand.
Reliable	• If you **specify** clearly how the model works and then **test** it thoroughly, you are much less likely to introduce errors.

Roles in modelling

It is helpful to define the different people involved in a spreadsheet project. For a simple model, these roles may be carried out by one or two people. As more people become involved with a model, it becomes more important to understand the different roles that people can take in the modelling.

Sponsor

The model sponsor is the person who requests that the model be built and ensures the required resources are available. The other part of the sponsor's role, not necessarily taken on by the same person, is to be the driving force behind the model. Agreement of the objectives of the model is the responsibility of the model sponsor.

Developer

The model developer translates the sponsor's ambition for the model into the actual spreadsheet model. Sometimes, different people will be involved in the separate stages to scope, specify and build the model.

User

There will be at least one user of the model and usually two, including the developer and sponsor. If other users are also involved, you need to decide when and how to introduce them to the model.

Reviewer

The reviewer is the person who tests the spreadsheet. In the chapter on testing models we discuss why this should be a different person from the model developer.

Risk factors in a model

Writing spreadsheets is a risky business because it is easy to produce a model that contains hidden errors, or is misunderstood.

The level of risk depends on the size and nature of the model and the way it is used. To understand how risky your model is likely to be, consider how many of these risk factors apply.

Complexity

A complex modelling problem is one in which the relationships between the inputs and calculated outputs are open to misunderstanding or are difficult to specify. When modelling is complex it is important to be very precise when explaining how the model works.

Size

Size is not necessarily the same as complexity of a model. Developing a large model will bring difficulties of its own. A large model will take longer to load, save and recalculate. The greater the size of a model, the more opportunity there is for errors in the coding.

Understanding of the problem

If, when you start modelling, your understanding of the problem is weak then there are additional risks in the model development. In these circumstances it is much more difficult to define the scope of the model or to estimate timescales. While the model is being developed, understanding of the problem will increase, but if you do not properly understand the problem until well into the development, the original design may be poor and errors may remain.

When the problem to be modelled is poorly understood, more attention is required at the initial stages of model scope and specification. More iterative processes of model development, such as the development of prototype models, may be appropriate.

Potential impact

Spreadsheet models can be used for a wide variety of purposes, some more important than others. A model might be developed to assist your own thought process or it might be used to make a critical decision. Errors in the model are possible in either case – but in the latter the consequences will be more serious.

Urgency of results

The more pressing the need for results from the model, the greater the risk of errors. Working under time pressure it is easy to make mistakes and there is less time available to spot errors or correct them.

Important and difficult decisions are frequently combined with time pressures. Modelling for such decisions is a very challenging job.

People involved

We discussed earlier four of the roles found in a modelling project. Some of these roles involve more than one person and other roles and sub-roles are possible. The more different people that are involved, the more opportunity there is for a lack of common understanding about what the model will do and how it will do it.

Applying the modelling life cycle

Spreadsheets are enormously flexible tools. The term ‘spreadsheet modelling’ covers a wide range of types of spreadsheet from simple one-off calculations through to complex pieces of software designed to reflect complicated interactions, with macros to automate the process. The principles of Spreadsheet Modelling Best Practice apply to all types of spreadsheet model, but require different interpretations for different models.

In our experience, one of the most difficult aspects of applying the ideas in this guide is deciding when and how to apply each part of the methodology. To get the most out of Spreadsheet Modelling Best Practice, you need to be able to apply it flexibly: understanding which parts are most relevant for the type of project you are working on.

Types of model

The importance of applying the six stages of the modelling life cycle can be matched to the six risk factors for models described above. At a simplistic level, the more of those risk factors that apply to your model, the more important it is to consider how you can use an understanding of the modelling life cycle to reduce the risk of errors in your model.

However, each of the model risk factors has a different effect on the stages of the modelling life cycle. For example, when the relationships in a model are highly complex, the specification stage is particularly important as a method for defining how the model will work.

To understand how to apply the modelling life cycle to different types of model, we find it useful to divide modelling projects into four major types. These types are:

- complex models, for which a number of the model risk factors apply, but there is adequate time for the development of an appropriate model;
- simple models, where the risk is less and the simpler techniques in the guide should be used;
- time-critical modelling, when the time available before results are required is short; and
- ill-defined problems, when it is more difficult to scope and specify the issues to be modelled.

In practice, a modelling project may contain features of more than one of these types.

Complex models

For complex modelling problems, but where there is a good understanding of the problem that is being modelled and there is adequate time for the development of an appropriate model, the six stages of the modelling life cycle can be considered, for the most part, as sequential steps that you can go through to develop and use a model. Applying each of the six stages helps to ensure the development of a robust model, reducing the risk of errors and getting the most from the modelling process.

Examples of types of models that fall into this category are long term financial planning or forecasting models that will be developed once and used many times.

Simple models

The extent to which the ideas for each stage need to be applied depends on the complexity of the modelling involved. For people who typically develop less complicated models (or models for which few of the risk factors apply) the full set of ideas described in this guide will seem burdensome.

To understand how to apply this guide to simpler models, bear in mind that:

- the specify chapter describes techniques of varying complexity to determine how your model works. When the relationships in the model are straightforward or very familiar the simplest of these, bubble diagrams, can adequately describe the working of the model; and
- the principles of good spreadsheet design apply equally to the simplest or most complex of models.

Examples of simple models are ones that take only a day of two to complete, or models that are used only by the developer for his or her own thought process.

Time-critical modelling

When the time available before some results are required from the model is short, developing a robust, error free model becomes more difficult. The model developer will come under pressure to skip the early stages of the modelling life cycle and dive straight into the build stage. Sometimes this pressure must be resisted because it leads to errors. But when there are important intermediate or final deadlines to be met a different approach to model development is required.

In this situation you need a method for model development that creates a model to meet the urgent deadlines, but still follows best practice guidelines and leads, eventually, to a complete, robust and error free model from which you can produce results with confidence.

Examples of types of models that fall into this category are models for particular financial transactions, bids, one-off decisions or any model that must meet a series of immovable deadlines.

The best approach is to consider the stages of the modelling life cycle very much as a cycle, where initial development of a model is then added to in a number of increments. When a model is required to produce intermediate results as well as go on to produce further results, this sort of approach is inevitable. Best practice development of this sort of model needs awareness of the advantages and the pitfalls of the different approaches. For example:

- if there are important and difficult time constraints, try to reduce the scope of the model (or the scope of the initial model);
- design the model, not just for the initial model development, but to allow additional features to be introduced later on. It is, however, inevitable that a model that evolves in a series of increments will have a weaker design than one planned and built in a single process;
- consider developing a simple model to meet the immediate deadlines, while a more thorough model is developed in parallel; and
- plan for the most appropriate time to test the model, based around when key results need to be produced. If you are forced to quote results from a model before it is tested, make sure that everyone understands that you are using an unfinished model that may contain significant errors.

Ill-defined problems

When the issues to be tackled by a model are ill defined and the objectives of the model are not agreed, it is difficult to scope and specify a model. To develop and agree a specification inevitably requires more than one attempt. When the problems which the model needs to address are ill-defined, the iteration in the model development becomes more important.

For a project of this sort you must be prepared to move back and forth between the different stages of the modelling life cycle, especially the scope, specify and design stages. Often, the understanding gained in this development process is of equal, if not greater, importance than the resulting model.

Examples of models that fall into this category are models of businesses undergoing dramatic change, for example due to changes in technology, or where the objectives of the model are potentially numerous, but not well understood.

Problems in this category are often best tackled by developing a series of prototype models. The specification chapter of this guide contains a description of the development of prototype models.

Summary of model types

Model type	Features of the modelling project	Applying the modelling life cycle
Complex models	• Part or all of the problem is complicated, unusual or a large model is required. • There is sufficient time available for model development. • Once developed, the model may be used many times. • It is important to develop a robust, error free model.	• Consider each of the six stages in turn to minimise the risk of errors.
Simple models	• The required model is relatively small and there are no particularly complicated parts. • The model developer is familiar with the type of model and has built similar models in the past. • The model developer and model user are the same person.	• An understanding of the modelling life cycle can improve your models without applying all of the guidelines described. • Use the simpler specification techniques. • Apply the design guidelines to all models, however simple.
Time-critical	• There is limited time available before an initial set of results is required. • The model will be used for a series of important decisions. • The model developer and model user are usually the same person.	• The modelling life cycle may be applied many times as the model continues to develop. • A formal specification sometimes lags behind development of the model, but is still an important process for defining the model and for testing it.
Ill-defined	• The problem to be modelled is particularly complicated or not well understood. • There are different possible applications for the model. • Once developed, the model may be used many times. • The understanding of the business problem is often as important an outcome of the modelling process as the finished model.	• The scope, specify and design stages of the model development are slower and more complicated. • Be prepared to move back and forth through the stages of the modelling life cycle as your understanding of the problem evolves. • Prototype models are often the best way to understand and communicate the problem.

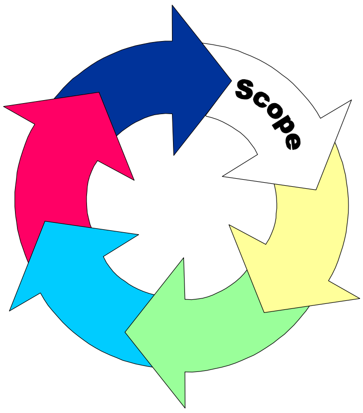
Figure 2: Summary of model types

In a number of the chapters of this guide, we will revisit these types of modelling project and consider how to apply the guidelines in that chapter to each type.

Chapter summary **The modelling life cycle**

- This guide is based around six stages in the life of a model: Scope, Specify, Design, Build, Test and Use.
- An understanding of the modelling life cycle, and application of the recommendations in this guide, depends on the type of model being developed. In this guide we consider four types of modelling project and how the life cycle differs between them.

3 Scope



The Scope of a model defines the objectives and boundaries of the model. In this chapter, we consider for your model:

- the objectives of the model, what it will (and will not) do;
- the appropriate level of complexity and what simplifying assumptions to make;
- data requirements for the model and when you need to produce the data;
- using workshops to build a common understanding of the model scope;
- the contents of the scope document; and
- how to estimate timescales for model development.

Objectives of the model

A model should have one or more clear objectives. Usually, if you have reached the scope stage, you have some idea of why the model should be developed. However, this idea may be ill-defined or there may be a number of additional alternative uses for the model.

In the scope stage of the project, you should clarify the model objectives and boundaries. The boundaries of the model define what the model will not do, whether because it would over-complicate the model or because there is not enough time available. You may decide that an optional feature of the model will not be included in the first version, but that the model should be designed to cope with it being added later on.

The model developer should drive this process by identifying the consequences of including alternative features in the model, as in the example model ‘shopping list’ below.

Model objective	**Comments**	**Include?**
Estimate the forecast bid price for the contract.	This is the main objective of the model.	Yes.
Forecast the effect on the company cash flow, P&L and balance sheet.	The model will have to make some assumptions about the performance of the rest of the company, based on our latest business plan: but it will be a useful tool for future planning.	Yes – include all three financial statements.
Model the operations – allowing us to flex productivity and raw material prices.	This could be useful, but would substantially increase the number of assumptions made each time we run the model and increase the time for development.	No – we will use the basic forecasts agreed at the meeting on 22 November.
Model the additional debt finance required to fund the contract.	This will need to be done before we sign any contracts, but we do not yet know what the structure of the debt will be.	No – but it is likely to be included in a later version of the model.
Model capacity restrictions at the plant.	This will help us estimate the capital programme required.	Yes.

Figure 3: Example model 'shopping list'

The appropriate level of complexity

The level of complexity included in the spreadsheet should be sufficient to meet the model's objectives, but no more. Over complex models are more difficult to build and maintain, and more likely to contain errors. A simple model, on the other hand, can be a valuable communication tool because more people can understand the business processes being modelled. There is a natural tendency for people to include excess detail, especially when they do not know exactly what they want.

To decide the appropriate level of complexity for the model, consider:

- What is the accuracy of the data you will be using?
- What is the appropriate unit of time to use (monthly, quarterly, annual)?
- How will the model outputs be used to make decisions and what level of accuracy for the results does this imply?

Whatever simplifying logical assumptions you decide to make, state them clearly. This gives other people involved in the modelling process time to consider your assumptions and challenge you if they want to, while there is still time to change your mind.

Try to make the distinction between logical assumptions, which determine the structure of the model and the formulae used, and input assumptions which are the estimates or forecasts to use in the model. At the scope stage, you are primarily concerned with the logical assumptions and the model structure.

Data requirements

The scope stage is the time to make sure that you understand, at least in broad terms, what data is required for the model and how you are going to obtain it. Difficulties in producing appropriate data are a common cause of delay in producing useful results from a model. If work is required obtaining or extracting the model data, it can be carried out in parallel to the model development provided that you identify the issue early enough.

Workshops

The scope stage is a time to collect opinions from the people involved in the model development, such as the model sponsor. Workshops are a good technique to use at this stage, especially when the model needs to reflect the opinions of a number of different people.

Use workshops to:

- agree the model objectives;
- discuss the trade off between the scope of the model and its effectiveness;
- resolve differences in the model requirements of the different people involved;
- agree the data that will be required for the model and responsibilities for producing it; and
- keep the relevant people informed about the development of the model.

When the problem is not well defined

When the problem to be modelled is ill defined, it is difficult to agree all of the model objectives. The initial scope can, however, help to bring together ideas about what is and is not practical or useful to model.

Useful things to do in the scope stage are to:

- produce a list of possible objectives for the model;
- highlight those which are particularly useful and some which are probably not realistic; and
- think about the areas of difficulty which need to be overcome before you can fully define the objectives of the model.

To define fully the objectives of the model, you may need to progress into the specification stage – to build up a more detailed understanding of how the model might work. Then you can be prepared to revisit the scope stage later to finalise on the model objectives.

“Getting off the fence”

When the demand for results from a model is particularly strong, the understanding developed during the scope stage can sometimes be used to make initial recommendations about the problem. By going through the process of scoping, the modeller has additional insight available. For example:

- an initial examination of the data might rule out some possible conclusions; and

- an understanding of the more likely conclusions of the model may help to understand which parts of the problem need to be modelled in greater detail and which can be simplified.

The process of model development has a number of objectives, but there is much more to it than just producing a completed model. It is the model developer’s responsibility not just to develop a model but to understand how the model can be used to draw conclusions and make decisions. It is not always possible to draw much in the way of conclusions at such an early stage, but throughout the process of model development the modeller should be aware of the uses of the model.

Contents of the scope document

The contents of the scope document will vary with the four types of modelling project that we introduced on page 2-8.

Model type	**Contents of the scope document**
Complex models	• State the agreed model objectives. • Describe the required complexity of the model. • Outline the data requirements. • Produce a work plan and timescales.
Simple models	• State the agreed model objectives. • Describe the required complexity of the model.
Time-critical	• State the agreed model objectives. • Describe the required complexity of the model. • Draw initial conclusions, where appropriate.
Ill-defined	• Focus on some of the model objectives and boundaries. • Make it clear when you will need to revisit the scope stage as your understanding of the model requirements develops.

Figure 4: Outcomes from a scope

Estimating timescales

For complex models, when there is a good understanding of the problem that is being modelled, one of the outcomes of the Scope stage should be an estimate of the time required to develop the model.

Estimating timescales is a notoriously difficult job and there is no perfect method of estimating the time required to develop a model. There are, however, some tips you can use.

The average proportions of time required for each stage of model development are shown in the table below, although it will vary for different projects. Note that a significant proportion of time is allowed for the initial Scope, Specify and Design stages for you start to build the model. A good rule of thumb is that the Build and Test stages require the same amount of time.

Stage	**Time (%)**
Scope	5
Specify	25
Design	10
Build	25
Test	25
Document	10
Total	**100**

Figure 5: Relative times for model development

When a model is being developed for a specific transaction, the time available to develop the model is often fixed, and it is the scope of the model that may have to change if time is short. For modelling projects of this type, a failure to develop the model in time is often caused by a failure to limit the scope of the model, either by restricting the model objectives or limiting the complexity of the assumptions.

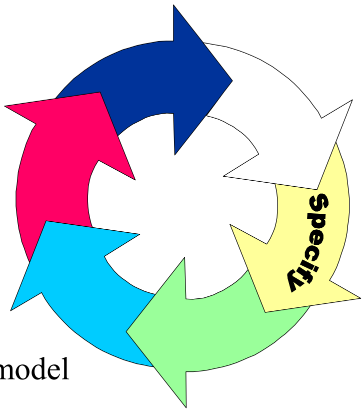
When your understanding of the problem to be modelled is ill defined, it is usually unrealistic to estimate the time required to develop a complete model. You need first to decide whether a spreadsheet model is the appropriate solution and what form the model should take.

There is little to be gained from trying to estimate the total time required to use the model. Instead, you may want to identify a limited list of core uses of the model and estimate the time required. If the model is successful, you will want to carry on using it.

Chapter summary **Scope**

- List the objectives of your model and the consequences for the model of deciding to meet each objective.
- State the boundaries of the model: what the model will **not** do.
- Keeping the model simple makes it a more useful tool for understanding and communicating the problem.
- Without the appropriate data – to the required level of accuracy – a finished model will be useless.
- When the problem to be modelled is not well understood, be prepared to revisit the scope stage once you have thought about the model specification.

4 Specify



The core of any spreadsheet is the definition of the formulae used to calculate the model's results. Specifying the formulae is the most important part of the creation of any spreadsheet model. For most people who write spreadsheets, this is done during the build stage, working out what each formula should be at the same time as typing it in.

We recommend that you separate the process of specifying the model calculations from actually building the model.

In this chapter, we:

- describe the benefits of separating the specify and build stages;
- discuss the process of defining outputs, calculations and inputs to the model; and
- introduce some techniques for defining model calculations and discuss when each should be used.

The benefits of model specification

The discipline of producing a model specification ensures that you produce a definitive statement of what the model should do, and how it will do it. It allows you to tackle the most difficult problems associated with the model before you have started to build it.

A model specification can be understood, discussed and challenged by all of the parties involved with the model. A successful specification establishes a common understanding of what the model will do.

A model specification makes model testing easier and much more effective. The specification provides a clear statement of what the model is doing. The model tester then has to check whether the finished model agrees with the written specification.

Failure to write a model specification leads to difficulties later on. An inadequate specification leads to:

- more time being spent building the model, because the issues that could have been resolved during the specification come back to haunt you;
- numerous late changes to the model logic – perhaps the greatest single cause of errors in completed models; and
- vague objectives for the model. It is easy to start with a small model that aims to solve one problem – and end up with a large model that solves all sorts of other problems, and solves them badly.

The process of model specification

Logically, all spreadsheet models should flow from inputs through calculations to the model results. The process of model specification should run the other way around, because it is model results that are most important and should determine the structure of the model.

Try to avoid letting the availability of input data determine the output that you produce. The model objectives should drive the results required, the results should drive the calculations and the calculations should drive the required inputs.

In practice, lack of available input data may restrict the results you can produce. If it is not possible to produce appropriate data you may need to restrict the model objectives, but do not allow the fact that relevant data is available to create additional functionality in the model that is not really required.

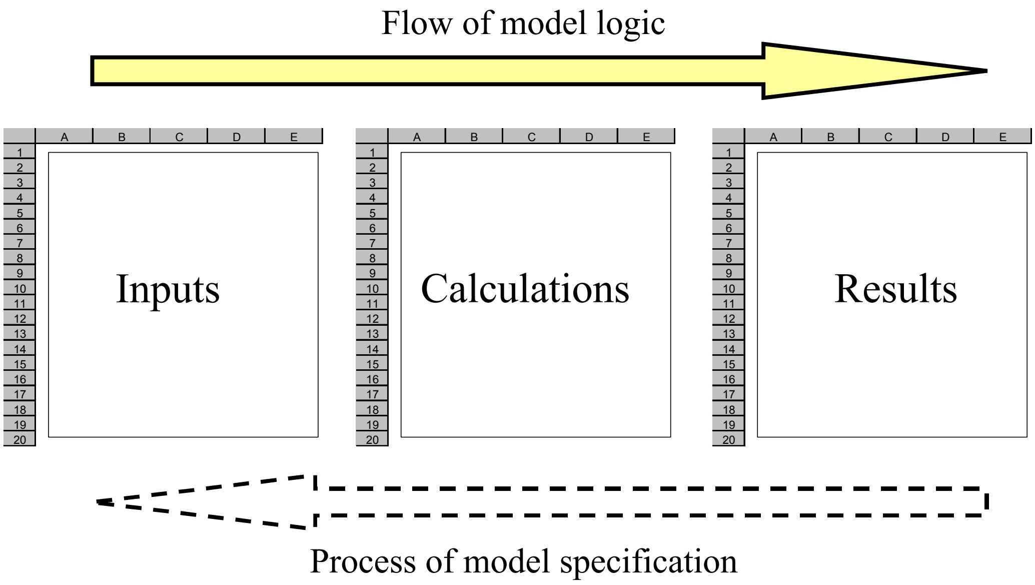


Figure 6: The process of model specification

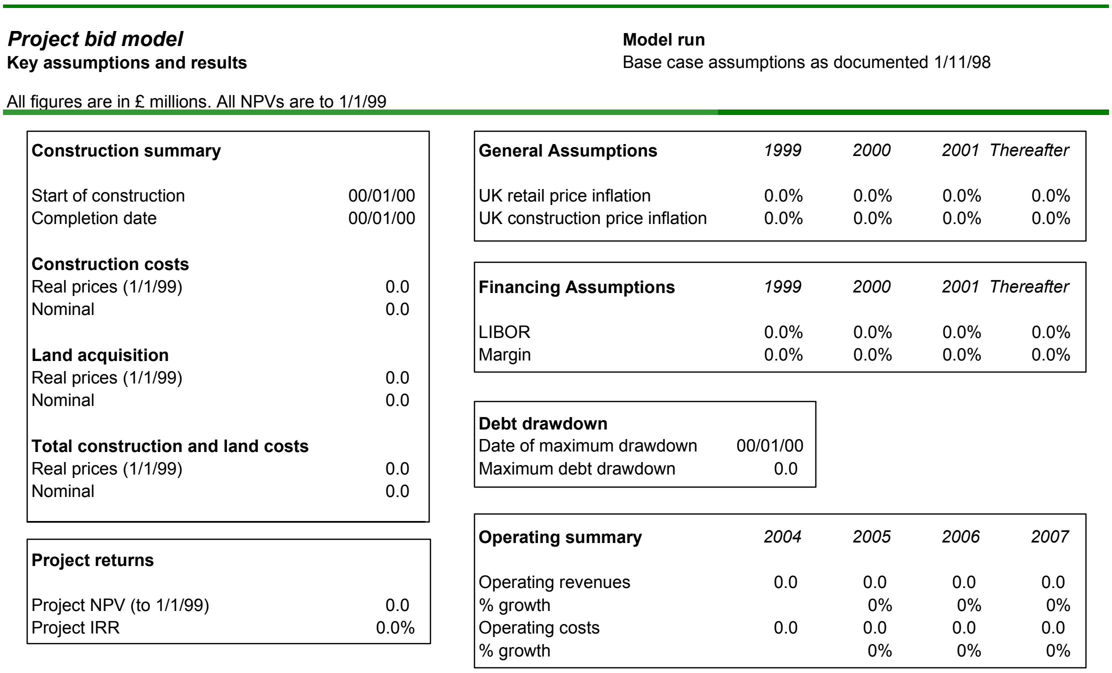
This chapter follows the process of model specification, from defining the outputs, through defining the calculations to defining the model inputs.

Defining model outputs

The first stage in the model specification process is to refine the broad output requirements that were defined from the scope stage into defined outputs that the model will produce.

Probably the best way to present the model outputs is to write outline reports using the spreadsheet package in which you intend to develop the model. Outline reports will look like the final model, but the numbers will simply be entered directly onto the report. These outline reports will give everyone involved with the model the best possible idea of what the final report will look like. Also, when the build stage begins, they provide a start for the final model itself.

Sometimes you can get the model sponsor to produce the outline reports.



Project bid model
Key assumptions and results

Model run
Base case assumptions as documented 1/11/98

All figures are in £ millions. All NPVs are to 1/1/99

Construction summary	
Start of construction	00/01/00
Completion date	00/01/00
Construction costs	
Real prices (1/1/99)	0.0
Nominal	0.0
Land acquisition	
Real prices (1/1/99)	0.0
Nominal	0.0
Total construction and land costs	
Real prices (1/1/99)	0.0
Nominal	0.0

Project returns	
Project NPV (to 1/1/99)	0.0
Project IRR	0.0%

General Assumptions	*1999*	*2000*	*2001*	*Thereafter*
UK retail price inflation	0.0%	0.0%	0.0%	0.0%
UK construction price inflation	0.0%	0.0%	0.0%	0.0%

Financing Assumptions	*1999*	*2000*	*2001*	*Thereafter*
LIBOR	0.0%	0.0%	0.0%	0.0%
Margin	0.0%	0.0%	0.0%	0.0%

Debt drawdown	
Date of maximum drawdown	00/01/00
Maximum debt drawdown	0.0

Operating summary	*2004*	*2005*	*2006*	*2007*
Operating revenues	0.0	0.0	0.0	0.0
% growth		0%	0%	0%
Operating costs	0.0	0.0	0.0	0.0
% growth		0%	0%	0%

Figure 7: Example outline report

This is a stage in which the model sponsor needs to be involved to agree what information is required in the model reports. Typical questions that need to be answered are:

- Who is going to use the model reports?
- What purpose is each model report going to serve?
- What is the appropriate level of detail to include on each report?

When you have established an outline for the model reports, you can consider the calculations needed to give you the required outputs.

Defining calculations: techniques for model specification

Developing the calculation rules that define the workings of a model can be difficult, especially when your understanding of the problem to be modelled is vague. That is why the process of defining the model specification is usually iterative, as more than one attempt is required before you get it right. You also need an approach that allows you to set out clearly how the model is to work: spreadsheets are good at presenting the layout of their calculations, but the detail of how they are working is hidden in the formulae.

This guide describes three techniques for developing a model specification: bubble diagrams, calculation tables and prototype models.

Bubble diagrams

A bubble diagram shows the logical flow of the model. They indicate the required inputs, and which calculations are required for each output from the model.

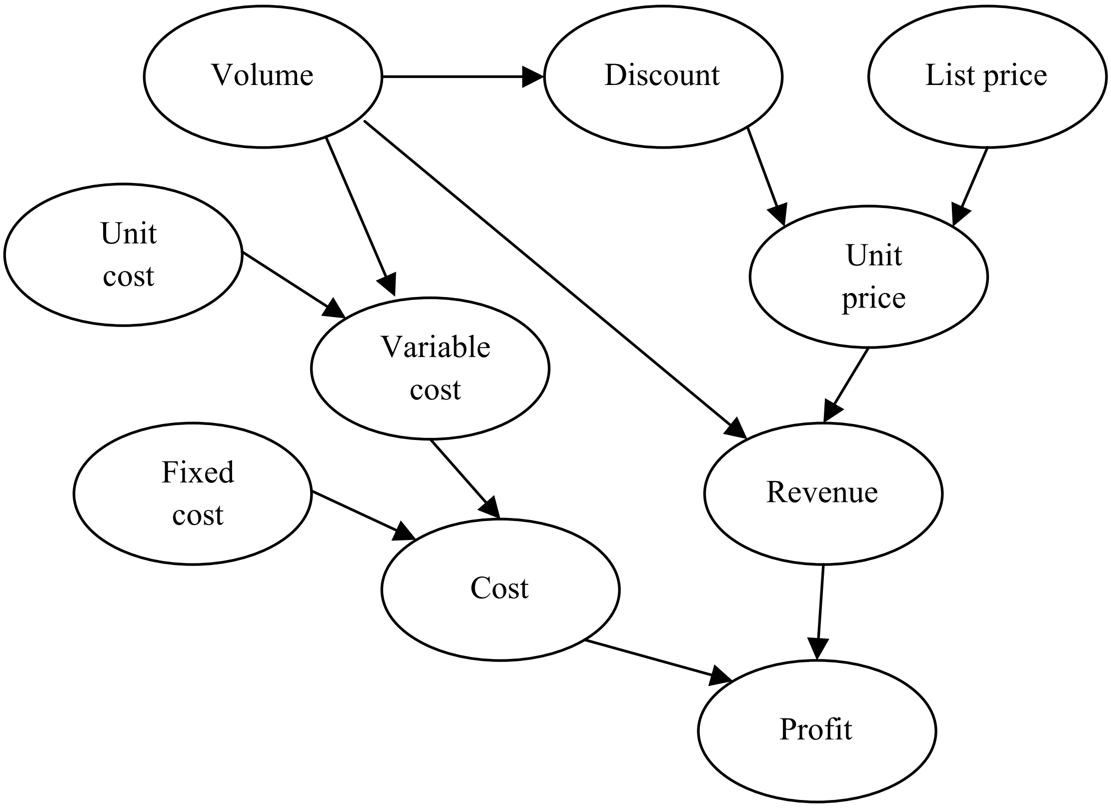


Figure 8: Example Bubble diagram (1)

This example illustrates how profit is calculated from a number of input assumptions. It shows, for example, that Unit price is calculated from the List price with an appropriate discount. The Discount is calculated from the Volume, presumably indicating a greater discount for bulk purchases. The bubble diagram does not however give precise details of how this discount mechanism works.

Bubble diagrams are fairly quick and easy to produce, and bubble diagrams can be updated as your thinking develops. Finished bubble diagrams make an excellent method of communication to explain the overall structure of a model. Bubble diagrams can be drawn in many different drawing packages; the diagrams in this guide were drawn using Microsoft Powerpoint 97.

Bubble diagrams can be constructed at a variety of levels, depending on the detail required. For example, a high level bubble diagram can be used for the overall structure of a model together with more detailed diagrams for particular parts of the model. The level of detail in a bubble diagram can also give an indication of the level of detail in the logical assumptions.

The second example bubble diagram below gives an example of a more detailed set of assumptions. In this example, inputs required to the model have been highlighted by shading the bubbles that represent inputs.

The major disadvantage of bubble diagrams is that they do not show the details of the calculations. It is sometimes possible to have a number of different, but equally valid, interpretations of what a bubble diagram is saying.

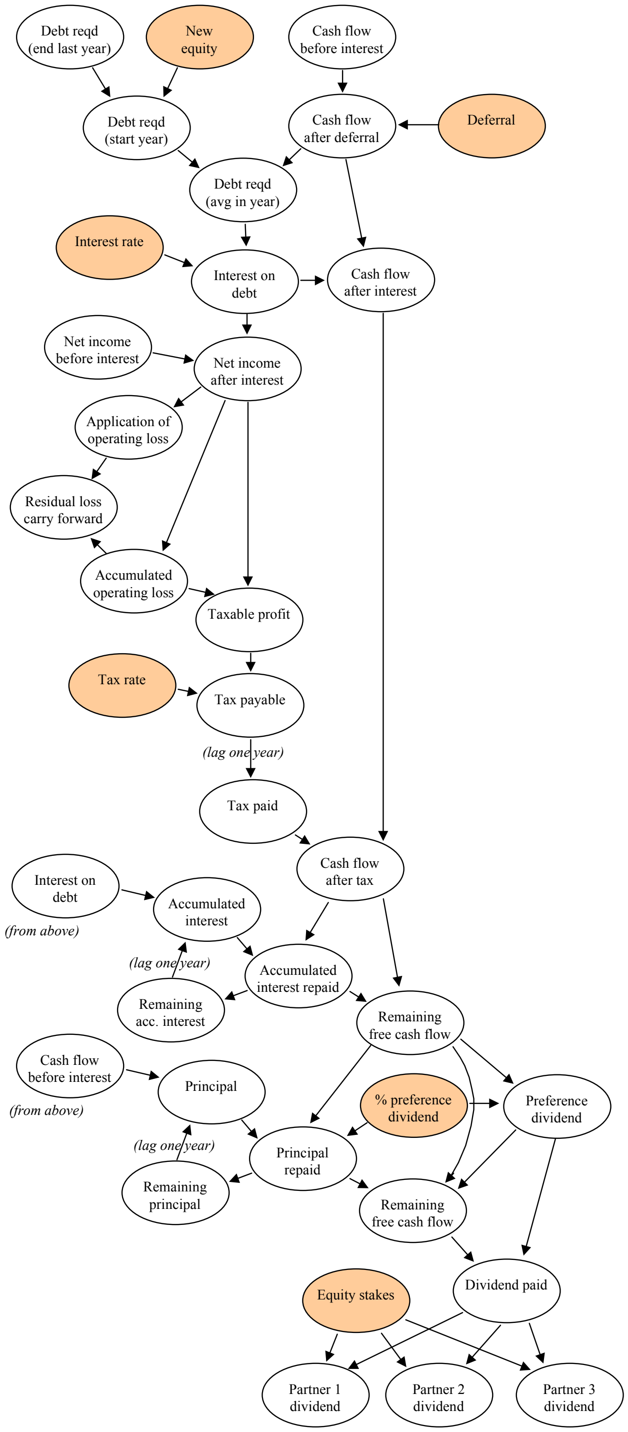


Figure 9: Example Bubble diagram (2)

Calculation tables

A calculation table is a detailed statement of the inputs and calculations in a model. It documents unambiguously the detail of how the model will work.

An example calculation table is shown overleaf.

Each item to be calculated has a description and a unique reference number. In this example, we are using the prefix 'Inp' (Inp10, Inp20, etc.) to indicate inputs to the model and the prefix 'Calc' (Calc10, Calc20, etc.) for calculations. This reference number is then used in the table to refer to each item when it is used in a calculation rule.

The calculation rule is a precise statement of the formula to be used in the final spreadsheet. It is generally written in a mixture of a spreadsheet formula and a natural language statement.

It should be possible to read the calculations from top to bottom down the page and understand the logic of the model. Therefore, all calculation rules should refer to reference numbers further up the table. This practice enforces a 'top to bottom' structure for the final spreadsheet model, which is good design practice.

Calculation tables are the most rigorous way of defining how the model will work. They are particularly useful when the relationships to be defined are complex or unusual. They are, however, time consuming to produce.

Ref No	Description	Ref No	Calculation rule	Units	Example	Notes
Inputs						
Inp10	Opening stock (year 1)			000's	150	
Inp20	Forecast sales			000's	180	
Inp30	Purchases			000's	60	
Calcs					(Year 1)	
Calc10	Opening stock		=IF		=IF	
			Year 1		Year 1	
			THEN		THEN	
		Inp10	Opening stock (year 1)	000's	I50	
			ELSE			
		Calc30	Closing stock (lag 1)	000's		
			=IF		=IF	
Calc20	Actual number sold	Calc10	Opening stock	000's	I50	
			+		+	Checks to see if there is sufficient stock to provide for the forecast sales
		Inp30	Purchases	000's	60 = 210	
			>		>	
		Inp20	Forecast sales	000's	180	
			THEN		THEN	
		Inp20	Forecast sales	000's	180	
			ELSE			
		Calc10	Opening stock	000's		
			+			
		Inp30	Purchases	000's		
			=		=	
Calc30	Closing stock	Calc10	Opening stock	000's	150	
			+		+	
		Inp30	Purchases	000's	60	
			-		-	
		Calc20	Actual number sold	000's	180 = 30	

Figure 10: Example of a Calculation table

Prototype models

A prototype model is an experimental model used to clarify calculation definitions prior to the development of the eventual model. As a technique for developing the model specification, it is most appropriate when:

- the problem to be solved is not well understood and the development of the model specification is a more iterative process;
- there are many possible objectives for the model and it is not clear how many of them will be achievable;
- it is important to communicate the progress being made in the development of the model by distributing part-developed models; and
- you expect to get useful feedback from a model sponsor who will make the time to look at the prototype models.

A prototype model can vary in scope and complexity from a small experiment to focus on one small part of the model development to a near complete version of the model. We have used prototype models to:

- clarify our understanding of the most difficult parts of the model specification;
- quickly try out alternative solutions; and
- carry out on-line presentations or distribute part-developed models to clients to get feedback on the best way to develop the final model.

A prototype model does not reduce the value of a more formal written specification, but once a prototype model has been developed, it is much quicker to write one. A prototype model can root out inconsistencies in a specification because when writing the prototype code it is impossible to gloss over any vague logical assumptions. Under some circumstances it is useful to produce initial results from a prototype model before the specification is complete.

The model sponsor must, however, understand the objectives of developing the model prototypes. There is a risk that a prototype is mistaken for a completed model when it was never designed to produce complete or reliable results.

As well as being a technique for developing a specification, a prototype model can be taken forward to develop the model design by experimenting with the structure, look and feel of the spreadsheet.

Although it is a useful technique, there are some risks associated with an over reliance on prototyping. For example:

- developed too far, a prototype model can have all of the problems of bad layout, lack of clarity and errors in formulae that are typically associated with skipping the preparatory stages of the modelling life cycle and diving straight into the build stage;

- although often easier to develop and use than calculation tables, prototypes do not set out clearly a benchmark for how the final model will work. Unless you then write a specification there is nothing to test the final model against; and

- developing thorough prototypes can be time consuming and still require you to start again to build the actual model.

When to use the specification techniques

The three techniques for developing specifications described in this guide each have strengths and weaknesses and are suited to different types of model development.

The table below summarises the key features of the three techniques.

	Bubble diagrams	**Calculation tables**	**Prototype models**
Experimentation	Good for setting out ideas and then re-working them.	Very rigid and time consuming to change.	Good for thinking through ideas and trying them out.
Speed	Quick to develop.	Slow to develop – but can speed up model building.	A quick way to start but can take up a lot of time on work that later needs to be repeated.
Communication	Excellent for communicating high level relationships.	Time consuming to understand – but all of the detail should be there.	It helps to be able to show a model – but the detail can be hidden in the formulae.
Testing	Can be misinterpreted, especially when the relationships are complex.	Provides a benchmark to test the final model against.	Provides little or no help for testing.
Rigour	Lacks detail.	Has all of the detail.	Should include the detailed calculations, but it is possible to overlook something important.

Figure 11: Key features of the specification techniques

In practice, most people will use a combination of techniques depending on the type of model they are developing. For many models a combination of bubble diagrams to show the overall structure and more detailed text or calculation tables for the complicated sections of the calculations is sufficient.

So which techniques are best applied to your type of model?

Model types

If we consider the four types of modelling project that we introduced on page 2-8, Figure 12 considers the most appropriate techniques to use.

Model type	Features of the specify stage
Complex models	• If the relationships in the model are complicated, calculation tables provide a rigorous definition of how the model will work. • If full calculation tables are too detailed an approach for your problem, you can provide detailed information of this type for the more complicated parts of your model. • You should agree the contents of your model specification document with the model sponsor as soon as possible, before you get too far into the model building.
Simple models	• The simpler the model, the shorter the specification document needs to be, but writing down the model objectives and the basics of how it will work is a valuable exercise.
Time-critical	• Full calculation tables will be impossible, but there are some short cuts that you can take. • Try to write down the detailed logical assumptions for the more complicated parts of the model, while providing sketchier information for the easier sections. • It is inevitable that the written specification will lag behind the latest set of changes to your model, but working to update the specification is still valuable for communication and model testing.
Ill-defined	• Prototyping and reworking is often the best way to tackle ill-defined problems. • Bubble diagrams can be a useful technique to communicate the outline of the logical assumptions, and they are easier to rework than the other specification techniques. • Be prepared to move back and forth through the scope, specify, design and build stages of the modelling lifecycle. • If your understanding of the problem develops, spreadsheet auditing software as discussed in Appendix A can be used to produce reasonable calculation tables from a prototype or part-built model.

Figure 12: Specification techniques for different model types

Defining model inputs

Once you have defined the model calculations, the inputs required in the model should be implicit. It is frequently worth considering in more detail what inputs are required and how they are to be sourced.

The table below is an example of a data input list. A data input list of this form is useful because it:

- establishes the level of detail required in the input assumptions at an early stage, in a form that is easy to communicate;
- highlights where special effort will be required in data collection – and assigns responsibility for producing the data; and
- in the status column, lists the type of estimate that will be used for each input assumption. This can help identify the inputs that will be candidates for sensitivity analysis.

Data input	**Format**	**Units**	**Frequency**	**Status**	**Validation**	**Source**	**Notes**
Interest rate	%, 2dp	%	One entry only.	Policy variable.	Must be > 0.	Finance dept.	The rate applied on any overdraft balances.
Sales	Numeric	000's units	One entry per month.	Forecast	Must be > 0.	Marketing dept.	
Price	Numeric	£	One entry per month.	Agreed estimate.	Must be > 0.	Marketing dept.	
Debtor period	Numeric	Days	One entry only.	Estimated from budget data.	Must be > 0; must be < model period.	Finance dept.	
Capital expenditure	Numeric	£000's	One entry per month.	Forecast		Capex model (auto transfer).	
Asset life	Numeric	Years	One entry only.	Policy variable.	Must be > 10.	Finance dept.	Calculated on a straight line basis.
Balance sheet – opening balance	Numeric	£000's	Start year only.	Firm figure.		Finance dept.	
Year end	Date		Start year only.	Firm figure.		Finance dept.	

Figure 13: Example of data input list

Hints and tips for model specification

Are constants constant?

Avoid using hard coded numbers in the definition of a formula. For example, the formula:

Sale price including VAT = Sale price excluding VAT × 1.175

assumes that the VAT rate is 17.5%. It works fine, until the VAT rate changes.

To avoid having to search through every formula in the model for calculations which assume the VAT rate, set up an input cell for the VAT rate and refer to that instead. For a single cell of this sort, it is useful to create a range name, say "VAT_rate", so instead of typing the constant you use the formula:

Sale price including VAT = Sale price excluding VAT × (1 + VAT_rate)

If there is any change to the input assumption, or you want to run a sensitivity, the constant can be changed throughout the model instantly. Even if the constant never changes, the new formula is much simpler to read and understand.

Real or nominal prices?

In financial models, a common problem is deciding whether to base monetary calculations on real or nominal prices. Real prices are prices of a particular base year and do not increase with inflation. Nominal prices, or money of the day, do include the effects of inflation.

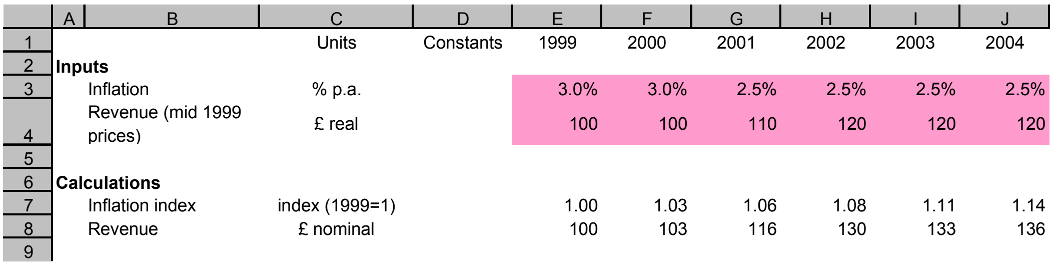
Real prices are frequently used for projections because it is possible to separate market trends from increases caused purely by inflation. For the same reason real prices are often used for summary results, particularly to calculate year on year changes in revenues or costs.

Using nominal prices is more accurate for any calculation involving monetary values over more than one time period. You should always use nominal prices for calculations of debt and interest, depreciation, tax or any stocks, such as assets and liabilities on a balance sheet.

Any moderately complex financial model must, therefore, include most or all of its calculations in nominal prices. However, real prices can be useful for input assumptions or for some results. So, the steps to go through are:

- inputs, often in real prices;
- conversion of any inputs into nominal prices;
- all required calculations carried out in nominal prices;
- conversion of results back to real prices, if required; and
- presentation of results.

To make conversion between real and nominal prices easy, it is a good idea to place the calculation of an RPI index in a clear place where it can be referred to quickly.



	A	B	C	D	E	F	G	H	I	J
1			Units	Constants	1999	2000	2001	2002	2003	2004
2	**Inputs**									
3		Inflation	% p.a.		3.0%	3.0%	2.5%	2.5%	2.5%	2.5%
4		Revenue (mid 1999 prices)	£ real		100	100	110	120	120	120
5										
6	**Calculations**									
7		Inflation index	index (1999=1)		1.00	1.03	1.06	1.08	1.11	1.14
8		Revenue	£ nominal		100	103	116	130	133	136
9										

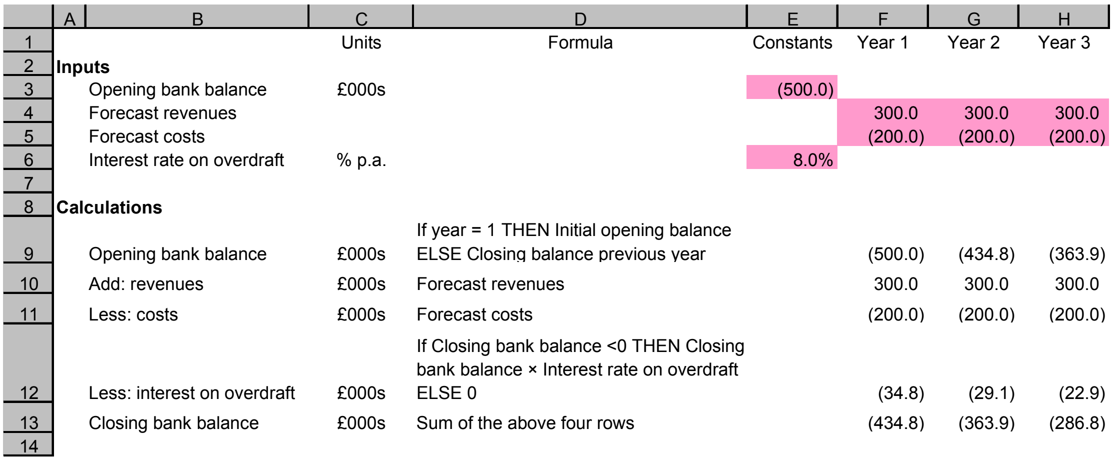
Figure 14: Example of real and nominal prices

Avoid circular references

Circular references should be avoided in model calculations. Frequently they are a sign of an error in the logic.

The best way to avoid a circular reference in your specification is to construct your calculation rules so that they always read from top to bottom and left to right: so that formulae always refer to cells above or to the left.

A common example of a circular reference arises when calculating interest on a bank overdraft. Consider the following example of the wrong way to do it.

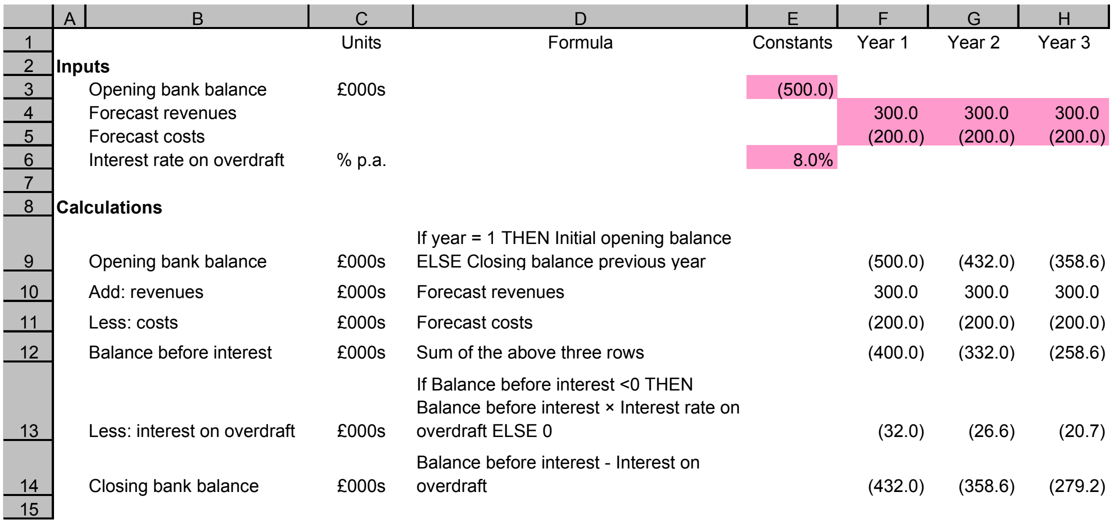


	A	B	C	D	E	F	G	H
1			Units	Formula	Constants	Year 1	Year 2	Year 3
2	**Inputs**							
3		Opening bank balance	£000s		(500.0)			
4		Forecast revenues				300.0	300.0	300.0
5		Forecast costs				(200.0)	(200.0)	(200.0)
6		Interest rate on overdraft	% p.a.		8.0%			
7								
8	**Calculations**							
9		Opening bank balance	£000s	If year = 1 THEN Initial opening balance ELSE Closing balance previous year		(500.0)	(434.8)	(363.9)
10		Add: revenues	£000s	Forecast revenues		300.0	300.0	300.0
11		Less: costs	£000s	Forecast costs		(200.0)	(200.0)	(200.0)
12		Less: interest on overdraft	£000s	If Closing bank balance <0 THEN Closing bank balance × Interest rate on overdraft ELSE 0		(34.8)	(29.1)	(22.9)
13		Closing bank balance	£000s	Sum of the above four rows		(434.8)	(363.9)	(286.8)
14								

Figure 15: Example bank overdraft: the wrong way

This calculation contains a circular reference because the interest on overdraft is calculated from the closing bank balance which is itself calculated from the interest. By default, Excel will refuse to calculate the spreadsheet above, although it is possible to allow it to iterate and produce the results above. For large models, iteration is too time consuming to be practical.

Without a circular reference, interest can be calculated like this.



	A	B	C	D	E	F	G	H
1			Units	Formula	Constants	Year 1	Year 2	Year 3
2	**Inputs**							
3		Opening bank balance	£000s		(500.0)			
4		Forecast revenues				300.0	300.0	300.0
5		Forecast costs				(200.0)	(200.0)	(200.0)
6		Interest rate on overdraft	% p.a.		8.0%			
7								
8	**Calculations**							
9		Opening bank balance	£000s	If year = 1 THEN Initial opening balance ELSE Closing balance previous year		(500.0)	(432.0)	(358.6)
10		Add: revenues	£000s	Forecast revenues		300.0	300.0	300.0
11		Less: costs	£000s	Forecast costs		(200.0)	(200.0)	(200.0)
12		Balance before interest	£000s	Sum of the above three rows		(400.0)	(332.0)	(258.6)
13		Less: interest on overdraft	£000s	If Balance before interest <0 THEN Balance before interest × Interest rate on overdraft ELSE 0		(32.0)	(26.6)	(20.7)
14		Closing bank balance	£000s	Balance before interest - Interest on overdraft		(432.0)	(358.6)	(279.2)
15								

Figure 16: Example bank overdraft: the right way

This calculation bases its interest calculation on an intermediate balance before interest is calculated – avoiding a circular reference.

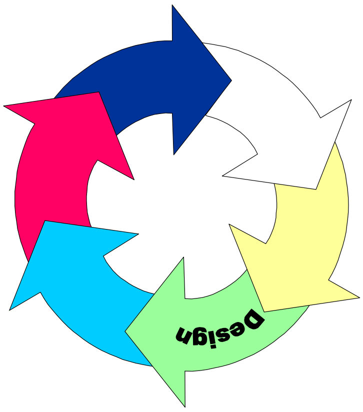
Sharp eyed readers will have noticed that these two methods produce different results, despite having the same inputs. The difference is caused by the compounding assumption implied by the two examples.

The first example (with the circular reference) assumes that interest is compounded instantly and interest on that interest starts to accrue immediately. The second example assumes that interest is compounded annually at the end of the year. The truth will depend on the type of debt being modelled, but is likely to be between these two extremes. All models contain simplifying assumptions; the calculation of interest is one that is often overlooked. It is good practice to explicitly state the assumptions you are using. In this case, the difference between these two examples illustrates that this is a significant assumption.

Chapter summary **Specify**

- Writing a model specification creates a definitive statement of what the model will do, and how it will do it.
- Writing a model specification makes building the model quicker and less prone to reworking, multiple objectives and errors.
- Specify the model outputs first, then the calculations and then the inputs.
- Testing a model is much more effective if you have a written model specification.
- Bubble diagrams are a good way to understand and communicate the general structure of calculations.
- Calculation tables are a rigorous way to specify a model – especially when the calculations are complicated and you have to get it right first time.
- Prototyping model calculations is especially useful when the problem to be modelled is not yet well understood.

5 Design



Good model design is one of the most striking features of a best practice spreadsheet model. A clear model design makes a model easy to use and understand. As a result, you are less likely to make errors while using the model and more likely to spot the mistakes you do make. A well designed spreadsheet is also easier for another person to pick up.

In this chapter we will:

- consider when a spreadsheet should be used for a modelling problem and when other software packages are more suitable;
- detail six golden rules of spreadsheet design that all models should follow;
- discuss methods for consolidating data in a spreadsheet; and
- consider when to use macros in the spreadsheet.

When to use a spreadsheet

Spreadsheet packages are good at numerical manipulation and have a wide range of financial and mathematical functions. It is easy to present calculations in a readable form and to mix text and graphical display. Spreadsheets are enormously popular, widely available and easy to use.

The flexibility of spreadsheets makes it possible to use them to tackle problems that would be more appropriately modelled with different software. Their availability and ease of use makes this an extremely common mistake.

Before you design a spreadsheet, make sure that a spreadsheet is the most appropriate tool for the job.

Excel 97 has a wide range of add-in functions that allow you to do a lot of unusual calculations. Many of these can be very useful, but if you find that you are using them a lot, it may suggest that you are better off using a specialist package. For example, if you are using a lot of the database functions, you probably should be using database software.

The table below compares the strengths and weaknesses of a number of different modelling packages.

Software type	Strengths	Weaknesses
Spreadsheets e.g.: Microsoft Excel, Lotus 123	• numeric manipulation; • financial functions; • user interface; • graphical reports; • easy to learn; and • time series modelling.	• handling large quantities of data; • multi-dimensional data; • systems with feedback or circularity; • looping and branching; and • can develop "black box" systems.
Databases e.g.: Microsoft Access	• handling large volumes of data; • user interface; • can develop "black box" systems; and • multi-dimensional data.	• complex calculations; • complex report structures; • graphical reports; and • time series modelling.
Statistical software e.g.: SAS	• handling large volumes of data; and • complex statistical functions.	• expensive; and • more difficult to learn.
Multi-dimensional packages e.g.: Oracle Financial Analyser	• multi-dimensional data; • handling large volumes of data; • "slice and dice" reporting; and • aggregation of data.	• specialised use; • more difficult to learn; • expensive; and • used more for information reporting than modelling.
System Dynamics packages e.g.: Vensim, Powersim	• systems with feedback or circularity; • "soft" variables such as staff morale; • multi-dimensional data; and • graphical representation of the model structure.	• producing financial statements; • difficult to understand and accept the processes; and • specialised skills required to develop and maintain.
Rules based packages e.g.: Applications Manager	• can develop "black box" systems; and • looping and branching.	• specialised use; and • more difficult to learn.

Figure 17: Comparison of modelling software

The six golden rules of spreadsheet design

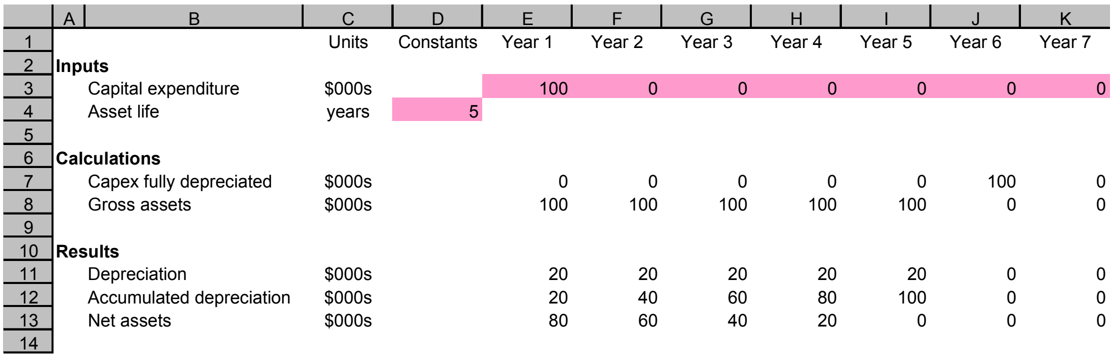
Following some simple rules of spreadsheet design makes models easier to understand, update and significantly reduces the risk of errors. Using consistent design rules across all people within an organisation makes it easier to hand over models.

Many of the chapters of this guide can be applied flexibly, depending on the type of modelling project. The rules of spreadsheet design apply to any spreadsheet model. It is arguable that for simpler models, where there is often less documentation available, discipline in the spreadsheet design is even more important to create easy to understand spreadsheet models.

Rule 1: Separate inputs, calculations and results

Inputs to a model should be placed on a separate part of the worksheet from calculations. It is also a good idea to place those results which are produced by the model in a separate place from intermediate calculations.

Separating inputs helps to avoid confusion in using and maintaining models. It reduces the risk that parts of the input data are overlooked or that calculations are overtyped with input figures. Inputs should be separated physically on the sheet, but can also be separated by clear labelling and colouring of the spreadsheet. The example below has input cells highlighted by colouring.



	A	B	C	D	E	F	G	H	I	J	K
1			Units	Constants	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7
2	Inputs										
3		Capital expenditure	$000s		100	0	0	0	0	0	0
4		Asset life	years	5							
5											
6	Calculations										
7		Capex fully depreciated	$000s		0	0	0	0	0	100	0
8		Gross assets	$000s		100	100	100	100	100	0	0
9											
10	Results										
11		Depreciation	$000s		20	20	20	20	20	0	0
12		Accumulated depreciation	$000s		20	40	60	80	100	0	0
13		Net assets	$000s		80	60	40	20	0	0	0
14											

Figure 18: Example sheet layout

When using multiple sheets, it can be a good idea to maintain a single sheet containing all of the model inputs in one place. So, whenever an input assumption needs to be changed, the user always knows where to go to look for it.

An alternative design is to place inputs at the top of each sheet which requires them. This keeps inputs fairly close to the calculations in which they will be used, while keeping them separate from the calculation.

Whatever design you use, avoid the temptation to place inputs amongst the calculations for which they are used, even if it is a temporary or last minute assumption. This technique leads to key input assumptions being overlooked.

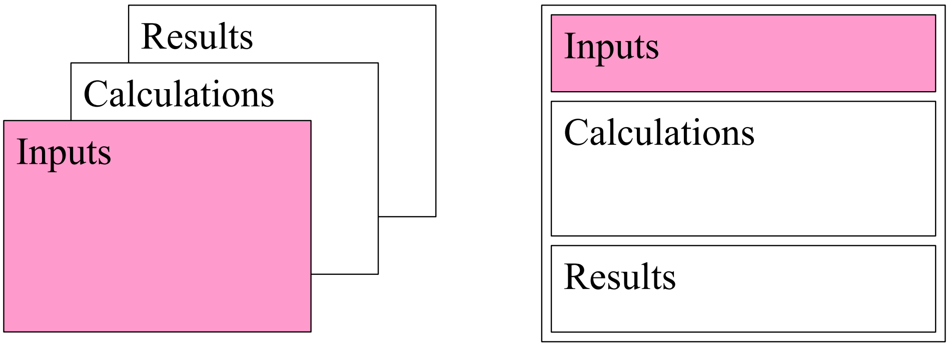


Figure 19: Multiple sheet design and… Single sheet design

An exception to this rule is when calculations are used specifically to make providing inputs to a model easier. For example, a checksum on a row of input values is more usefully placed in the input section of the spreadsheet than anywhere else.

Rule 2: Use one formula per row or column

As far as possible, formulae should be written so that a single formula can be copied across the entire row of calculations, or down the entire column. Designing a spreadsheet with one formula per row leads to a number of practical benefits, such as:

- quicker development, because every formula can be written in a single column, then copied across en masse;
- effective testing, especially when you use formula maps of the type recommended in the test chapter of this guide;
- robust updating and maintenance, because every time you change a formula in a cell you know that it should be copied across the rest of the row. One of the most common causes of errors in formulae is when a change is made, but old versions of the formula are left lurking behind; and
- straightforward documentation, in a specification which has one definition for each row of the final model.

For some problems, producing a design with only one formula per row requires some careful thought. But the benefits of the design far outweigh the extra work required.

Using IF statements

Not all calculation rules are automatically the same for each time period. For example, in a quarterly model some costs may be calculated only once a year. This problem can be solved using an IF statement. For example, the formula

IF (quarter = 4, Cost calculation, 0)

can be safely copied across every column.

When an opening balance is calculated in a different way for the first time period, an IF statement also provides an appropriate solution. For example:

IF (time period = 1,

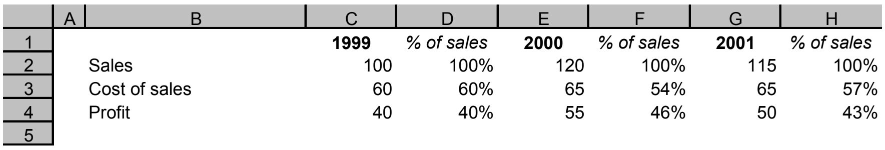
Opening balance provided as an input to the model,

Closing balance calculated for previous period)

will extract an input value for the first time period, but calculations for subsequent periods. This provides a simple solution without either using more than one formula in a row or mixing inputs and calculations.

Intermediate totals

Another inconvenience when using a single formula per row is trying to include sub-totals, or other intermediate calculations. This is commonly approached with a design like the one below.

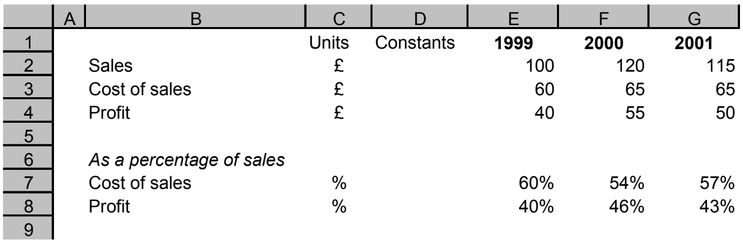


	A	B	C	D	E	F	G	H
1			**1999**	*% of sales*	**2000**	*% of sales*	**2001**	*% of sales*
2		Sales	100	100%	120	100%	115	100%
3		Cost of sales	60	60%	65	54%	65	57%
4		Profit	40	40%	55	46%	50	43%
5								

Figure 20: Example of poor design

Unfortunately this breaks up the formula in each row. If the definition of cost of sales is changed, the updated formula has to be copied across separately for each year which is awkward and may lead to an error.

A much better design is to produce two separate blocks of calculations.



	A	B	C	D	E	F	G
1			Units	Constants	**1999**	**2000**	**2001**
2		Sales	£		100	120	115
3		Cost of sales	£		60	65	65
4		Profit	£		40	55	50
5							
6		*As a percentage of sales*					
7		Cost of sales	%		60%	54%	57%
8		Profit	%		40%	46%	43%
9							

Figure 21: Example of better design

Changing time periods

Models are sometimes required to produce output in time periods of increasing length, for example to produce quarterly results for the first few years and annual results thereafter. Designing a spreadsheet to produce output in this way is difficult. For example:

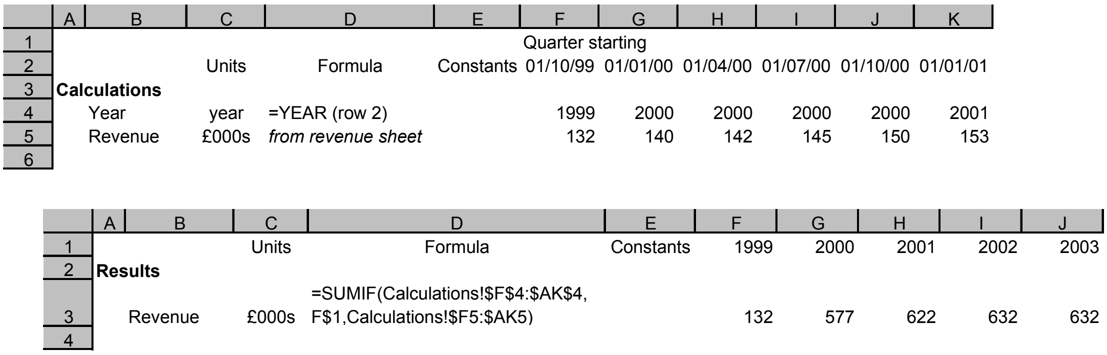
- any assumption that involves a time lag, such as stock movements or tax payments, may have a different time delay in different parts of the model; and

- at the point when the frequency of time periods changes, it is easy to confuse the different assumptions required and make mistakes.

If you are faced with designing a model in this way, the first step we recommend is to challenge whether changing time periods is actually necessary in the model. Are quarterly time periods for the first few years really necessary? or would the model be better replaced with a short term budget planning model for the first few years and a separate long term planning model?

If a change in time periods is necessary then, unless you are particularly worried about the overall size of the model, the best solution is to design the model around the shortest time period required. In the example we have been considering this would mean using quarters throughout the life of the model. If you are required to produce output in annual terms, you can perform the calculations quarterly, then consolidate the output to produce results annually. This will ensure that the underlying model has consistent logical assumptions across all of the time periods.

The example below consolidates quarterly calculations into a report that shows annual results. It uses a single unique formula in each row of the output.



	A	B	C	D	E	F	G	H	I	J	K
1						Quarter starting					
2			Units	Formula	Constants	01/10/99	01/01/00	01/04/00	01/07/00	01/10/00	01/01/01
3	**Calculations**										
4		Year	year	=YEAR (row 2)		1999	2000	2000	2000	2000	2001
5		Revenue	£000s	*from revenue sheet*		132	140	142	145	150	153
6											

	A	B	C	D	E	F	G	H	I	J
1			Units	Formula	Constants	1999	2000	2001	2002	2003
2	**Results**									
3		Revenue	£000s	=SUMIF(Calculations!$F$4:$AK$4, F$1,Calculations!$F5:$AK5)		132	577	622	632	632
4										

Figure 22: Example of consolidating quarterly calculations to an annual report

If a change in time periods is necessary and you are concerned about the size implications of using the shortest time period throughout the model, the best approach is probably to separate the calculation blocks used for each time period, so you have one worksheet doing calculations for the first few years and a completely separate calculation for the later years.

Whatever method you use, try to avoid what is sometimes the most tempting design: to have two different formulae in each row of the spreadsheet, the formula changing at the point where the time periods change. This method leads to a potential error every time that you change a formula in the spreadsheet and copy it across the row.

Rule 3: Refer to the left and above

A well designed spreadsheet can be read like a book: from front to back, top to bottom and left to right. This makes the spreadsheet easier to understand and reduces the risk of introducing a circular reference to the calculation.

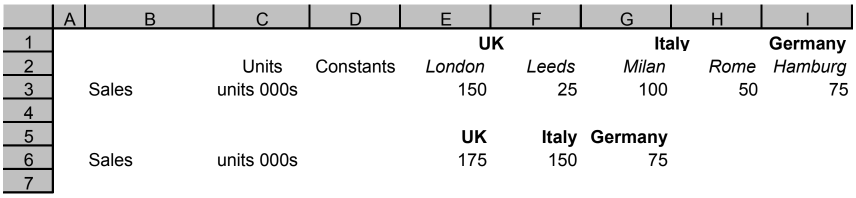
For example, if your model has a different time period in each column, each cell reference in a formula could be:

- to an input or calculation further up the same column;
- to an input or calculation on an earlier sheet in the model; and
- to a calculation further down the sheet but in an earlier column, for example referencing the closing balance calculated in the previous time period to use as the opening balance in this time period.

Rule 4: Use multiple worksheets…

1. For ease of expansion

Consider the following example for monitoring sales from a company's various European offices.

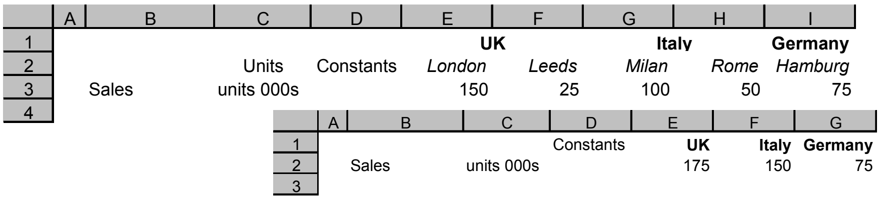


	A	B	C	D	E	F	G	H	I
1					**UK**		**Italy**		**Germany**
2			Units	Constants	*London*	*Leeds*	*Milan*	*Rome*	*Hamburg*
3		Sales	units 000s		150	25	100	50	75
4									
5					**UK**	**Italy**	**Germany**		
6		Sales	units 000s		175	150	75		
7									

Figure 23: Example of poor use of worksheets

If the company acquires a third UK office, it would be tempting to insert an additional column between E and F to make space for it. But to do so would break up the summary table of sales by country. It is possible to just insert a block for rows 1 to 3, but it is easy to forget: especially when there are more rows, and the summary table is not always visible on screen.

A far better design, more robust to future changes, is to use separate worksheets whenever there are different headings required.



	A	B	C	D	E	F	G	H	I
1					**UK**		**Italy**		**Germany**
2			Units	Constants	*London*	*Leeds*	*Milan*	*Rome*	*Hamburg*
3		Sales	units 000s		150	25	100	50	75
4									

	A	B	C	D	E	F	G
1				Constants	**UK**	**Italy**	**Germany**
2		Sales	units 000s		175	150	75
3							

Figure 24: Example of better use of worksheets

In the days when not all spreadsheet packages could produce multiple worksheets, it was common practice to use a diagonal layout for calculations of this sort. Using multiple sheets resolves the same problem, but creates a model that is much easier to navigate.

2. For repeatable blocks

When a model contains sections which can usefully be repeated, such as a model of a company with a number of similar business units, the best way to present it in a spreadsheet is to use one worksheet for each repeatable block. This approach will allow for:

- quicker development, by creating one standard sheet which you can then copy a number of times;
- easier updating, by grouping a number of sheets in the model you can update all of the blocks of the model together, which would be impossible if the blocks were all kept on the same sheet;
- quicker testing, by comparing the formulae in each of these sheets and making sure that they are the same. The spreadsheet auditing software that we discuss in Appendix A will do this job automatically; and

- use and reuse, by developing a number of separate sheets that can be reused in future models.

When the model for a number of subsidiaries in a business are similar, but not quite the same, it is still a good idea to use comparable worksheets for each subsidiary. When some subsidiaries require more detail, it is easy to develop a worksheet for the most complex example and then blank out the relevant sections for other subsidiaries. Do not delete any rows or columns, so that the common elements on the different worksheets can still be edited together.

Rule 5: Use each column for the same purpose throughout the model

However many worksheets you use, it is good practice to always use the same layout for columns on all worksheets.

For a financial model, an example column layout might be:

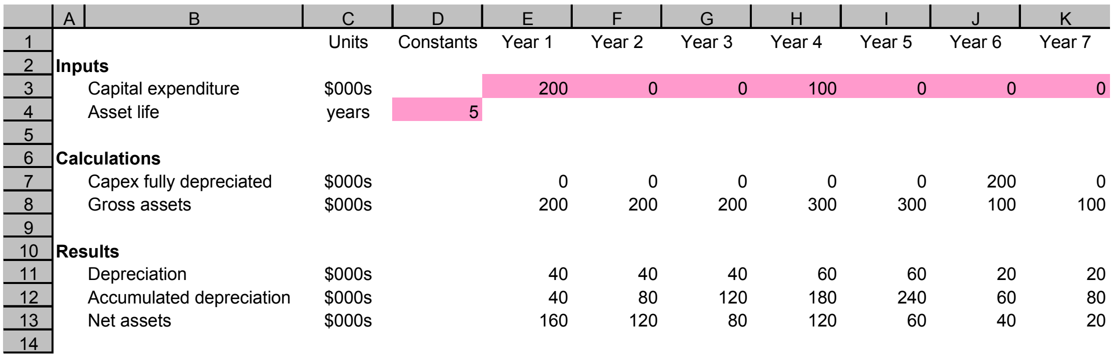
Columns A. and B. Labels

Column C. Units: a definition of the units used for the values in that row

Column D. Constants: inputs or calculations which apply irrespective of the time period

Column E. First time period

Columns F. onwards. Subsequent time periods.



	A	B	C	D	E	F	G	H	I	J	K
1			Units	Constants	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7
2	Inputs										
3		Capital expenditure	$000s		200	0	0	100	0	0	0
4		Asset life	years	5							
5											
6	Calculations										
7		Capex fully depreciated	$000s		0	0	0	0	0	200	0
8		Gross assets	$000s		200	200	200	300	300	100	100
9											
10	Results										
11		Depreciation	$000s		40	40	40	60	60	20	20
12		Accumulated depreciation	$000s		40	80	120	180	240	60	80
13		Net assets	$000s		160	120	80	120	60	40	20
14											

Figure 25: Example use of columns

Always using the same column for the same purpose makes the model easier to build and reduces some of the risks of errors. For example:

- if January 2001 is always in column X then you know that every formula that refers to January 2001 will also refer to column X;

- if constants are in column D, then nearly every reference to column D should have an absolute reference of the form $D1. If a formula contains a relative reference such as D1, it is probably an error; and

- except in very unusual circumstances, a formula in a given time period may refer to the constants column, the same time period and possibly the previous time period. A reference to any other time period is probably an error.

Rule 6: Include a documentation sheet

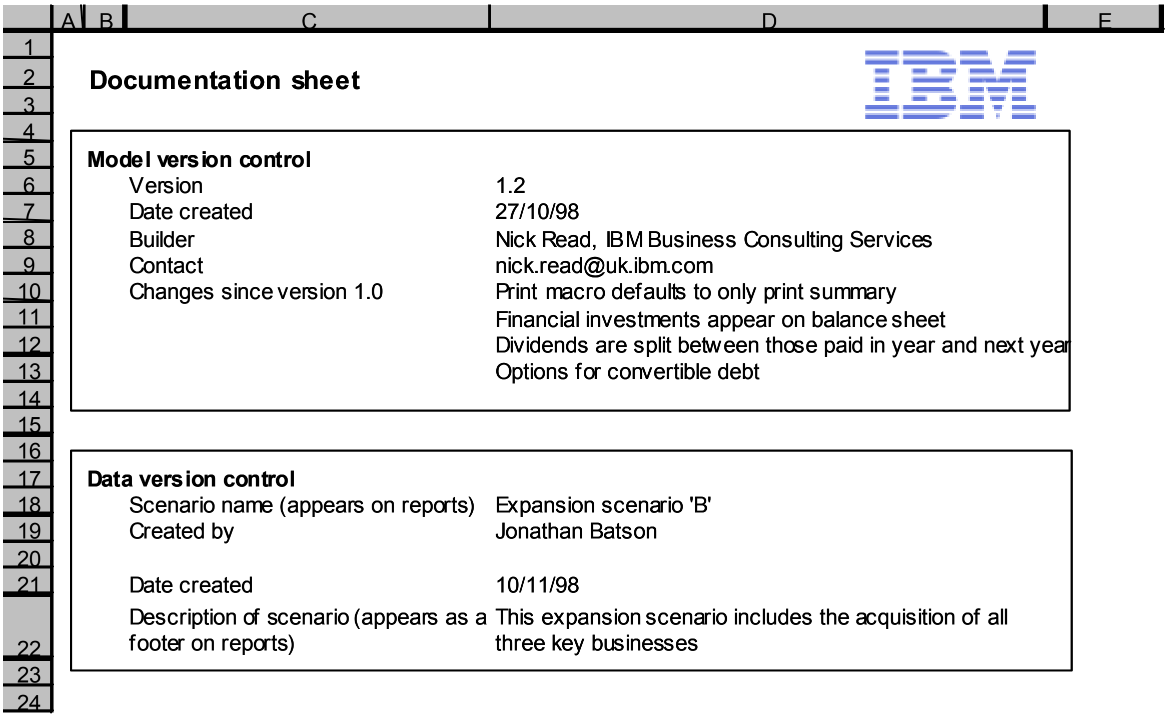
Any model should contain some internal documentation. To make it noticeable, it is a good idea to set aside the first sheet of the spreadsheet as a documentation sheet, used purely for model control.

The documentation should include:

- a short description of the model's purpose;
- who built the model;
- how to contact the person responsible for the model; and
- the model version number and when it was written.

Depending on the model, other useful items to include on the documentation sheet are:

- details of the data which is currently in the model;
- some brief instructions, describing the layout of the model or how to use it;
- a list of recent changes to the model; and
- a summary of key logical assumptions in the model.



Documentation sheet

Model version control

Version	1.2
Date created	27/10/98
Builder	Nick Read, IBM Business Consulting Services
Contact	nick.read@uk.ibm.com
Changes since version 1.0	Print macro defaults to only print summary Financial investments appear on balance sheet Dividends are split between those paid in year and next year Options for convertible debt

Data version control

Scenario name (appears on reports)	Expansion scenario 'B'
Created by	Jonathan Batson
Date created	10/11/98
Description of scenario (appears as a footer on reports)	This expansion scenario includes the acquisition of all three key businesses

Figure 26: Example documentation sheet

Hints and tips for model design

While not being appropriate to all spreadsheets, the design ideas in this section can all be useful ways to make your model easier to understand and use.

Colour coding

Colour coding is a useful technique to make spreadsheets easier to use. If you use dark colours for text then black and white printouts will be largely unaffected, but the difference will be visible on screen. Using pale colours for shading cells will be visible on printouts, but may become indistinct on photocopies or faxes.

Examples of using colour coding include:

- separating inputs from calculations, by making all of the cells requiring inputs the same colour;

- separating types of inputs, such as by making the inputs required from the finance department one colour and those required from the marketing department another colour;

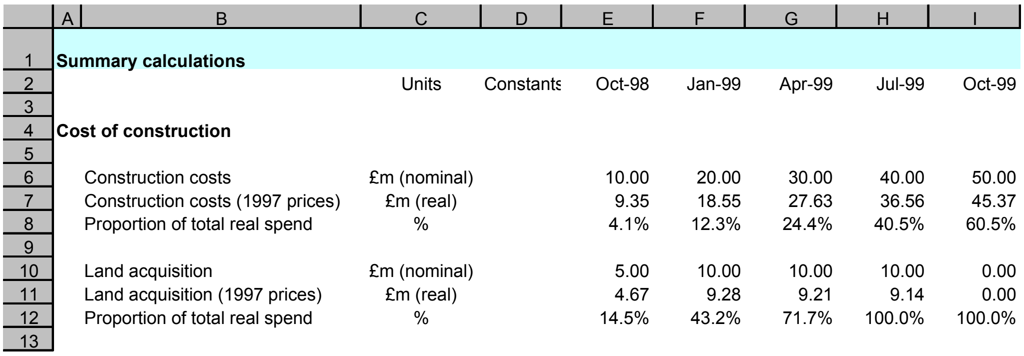
- highlighting cells which are linked to external spreadsheets; and

- highlighting the cells modified when you made a particular change to the model, making it much easier to track changes or, if necessary, reverse the change at a later date.

Units

A units column on every worksheet will tell you what is expected in each and every input cell and what is being presented in calculations. It is particularly useful when different units are being used in different places. For example, if the price per widget is input in dollars, but the annual turnover is presented in millions.

If the model uses a mixture of real and nominal monetary values, make sure the units column always specifies which is required. To avoid confusion, always make it clear in prices of which year real inputs are required.



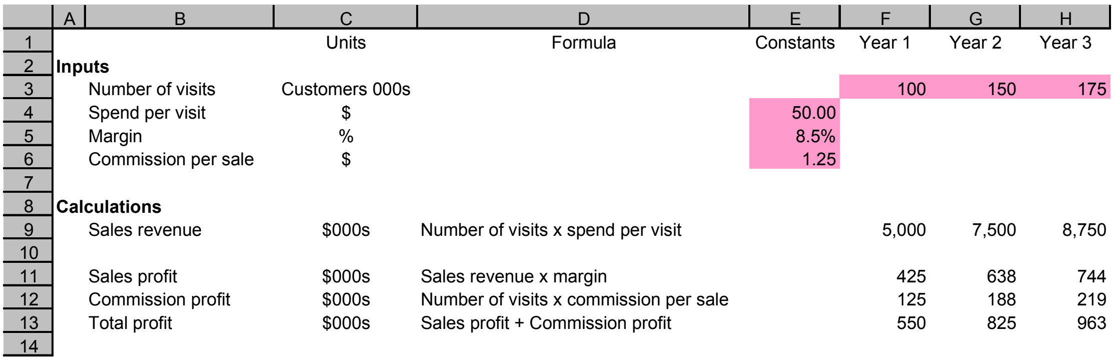
	A	B	C	D	E	F	G	H	I
1	Summary calculations								
2			Units	Constants	Oct-98	Jan-99	Apr-99	Jul-99	Oct-99
3									
4	Cost of construction								
5									
6		Construction costs	£m (nominal)		10.00	20.00	30.00	40.00	50.00
7		Construction costs (1997 prices)	£m (real)		9.35	18.55	27.63	36.56	45.37
8		Proportion of total real spend	%		4.1%	12.3%	24.4%	40.5%	60.5%
9									
10		Land acquisition	£m (nominal)		5.00	10.00	10.00	10.00	0.00
11		Land acquisition (1997 prices)	£m (real)		4.67	9.28	9.21	9.14	0.00
12		Proportion of total real spend	%		14.5%	43.2%	71.7%	100.0%	100.0%
13									

Figure 267: Example of units column

Formulae in English

For simple models, with little formal documentation, it can be useful to include an extra column containing natural language ‘translations’ of the formulae in the model.

This helps to make the calculations in a model easy to understand, makes important logical assumptions more transparent to anyone using the model and makes it possible for an independent tester to check the coding in the model.



	A	B	C	D	E	F	G	H
1			Units	Formula	Constants	Year 1	Year 2	Year 3
2	**Inputs**							
3		Number of visits	Customers 000s			100	150	175
4		Spend per visit	$		50.00			
5		Margin	%		8.5%			
6		Commission per sale	$		1.25			
7								
8	**Calculations**							
9		Sales revenue	$000s	Number of visits x spend per visit		5,000	7,500	8,750
10								
11		Sales profit	$000s	Sales revenue x margin		425	638	744
12		Commission profit	$000s	Number of visits x commission per sale		125	188	219
13		Total profit	$000s	Sales profit + Commission profit		550	825	963
14								

Figure 27: Layout example with formulae described in English

When the model development is complete, the column containing the formula can be safely hidden, but not deleted to keep it as a record of what the model is attempting to do.

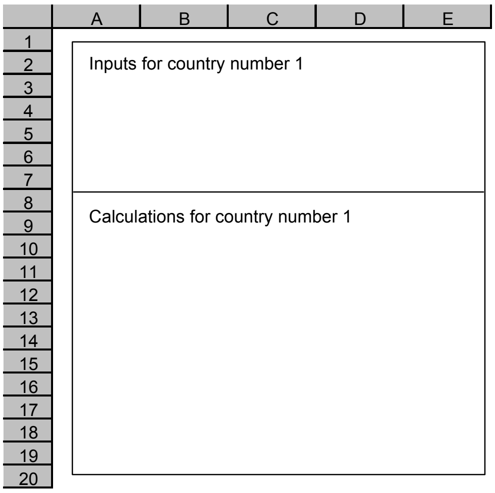
Consolidating and connecting worksheets

There are three common methods for consolidating data or connecting worksheets:

- using multiple sheets within a single spreadsheet;
- using separate spreadsheets with external links between them; and
- using macros.

The most appropriate solution will depend on the size and complexity of the model and the experience of the model developer and user. Consider this example.

You need to consolidate some information for a business that operates in 24 countries. You have a standard template for each country that contains the relevant information in a single block of calculations on a worksheet. To convert this information into a summary of the whole business, there are a number of different methods you could use.



Inputs for country number 1

Calculations for country number 1

Figure 28: Example sheet to be consolidated

Method 1

Probably the most obvious method is to organise the calculations from the 24 countries onto 24 worksheets in a single spreadsheet model. It is then straightforward to include one additional consolidation sheet that contains formulae to sum through the 24 sheets to calculate the totals.

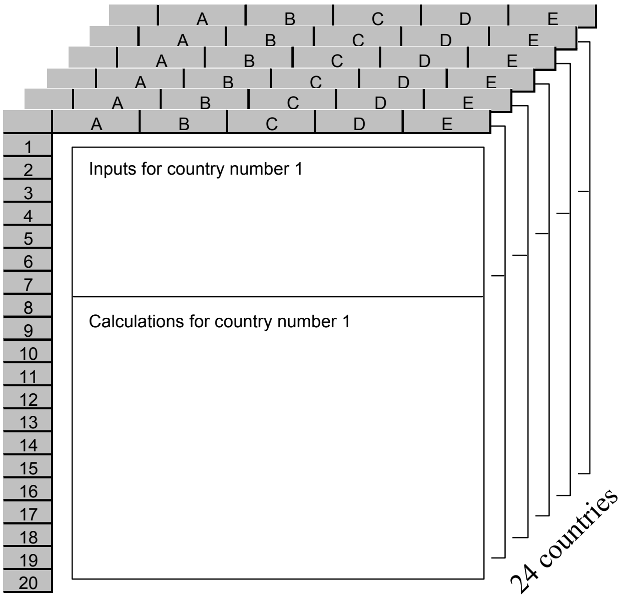

A
B
C
D
E
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Inputs for country number 1
Calculations for country number 1
24 countries

Figure 29: Consolidation with multiple sheets

The advantages of this method are that:

- it is easy to understand, with all of the calculations and data visible;
- it is simple to build; and
- inserting an additional country will be fairly easy, provided that you do not want to insert the country at the beginning or the end.

However, by using this method you will:

- create quite a large model, especially if there is a large quantity of information to consolidate for each country; and
- need to find a method to introduce the inputs from each country into the consolidation model.

Method 2

If the inputs from each country have been provided in the worksheet template above, an alternative method is to use externally linked spreadsheets.

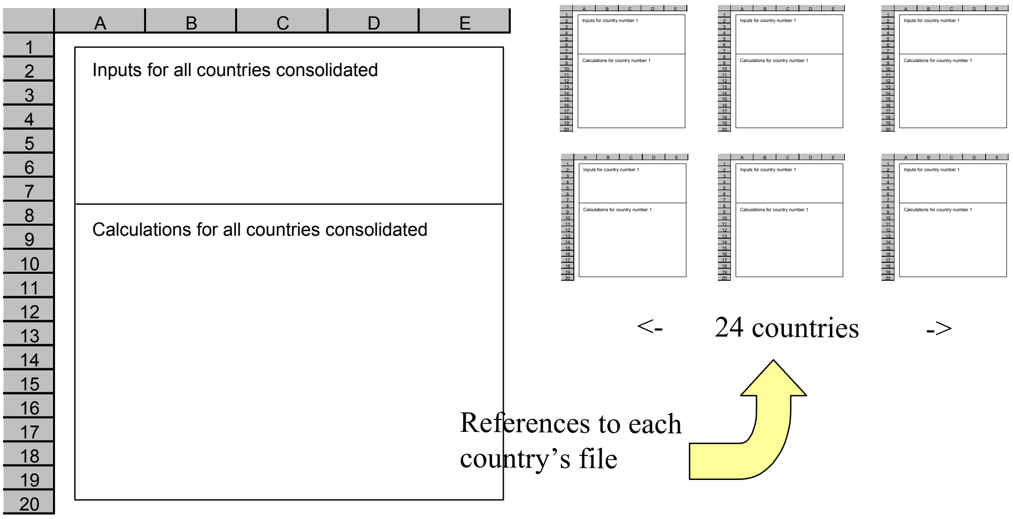

A
B
C
D
E
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Inputs for all countries consolidated
Calculations for all countries consolidated
Inputs for country number 1
Calculations for country number 1
<- 24 countries ->
References to each country's file

Figure 30: Linking external files

Using this method, each formula in the consolidation contains a reference to each country's separate spreadsheet. A typical formula might look something like:

= 'C:\Countries\[Austria.Xls]Results'!$C$3 + 'C:\Countries\[Belgium.Xls]Results'!$C$3 +
'C:\Countries\[France.Xls]Results'!$C$3 + … (21 more countries)

The advantages of this method are that:

- it is reasonably easy to understand;
- none of the spreadsheet files will be particularly large; and
- it provides a method of extracting values from a number of separate spreadsheets.

Disadvantages are that:

- when a large number of files are involved, the formulae become long and unwieldy. For 24 countries this method is inappropriate;
- it is awkward and time consuming to add an extra country;
- it is easy to introduce errors by making changes the individual country's files and failing to update the links; and
- it is likely to be slow to update the links.

Method 3

The third major method is to use macros to automatically copy and consolidate the individual countries' spreadsheets. The consolidation model contains a list of all of the file names for the individual countries. A macro will then go through the list of files, extract the data for that country and add it to the consolidation table.

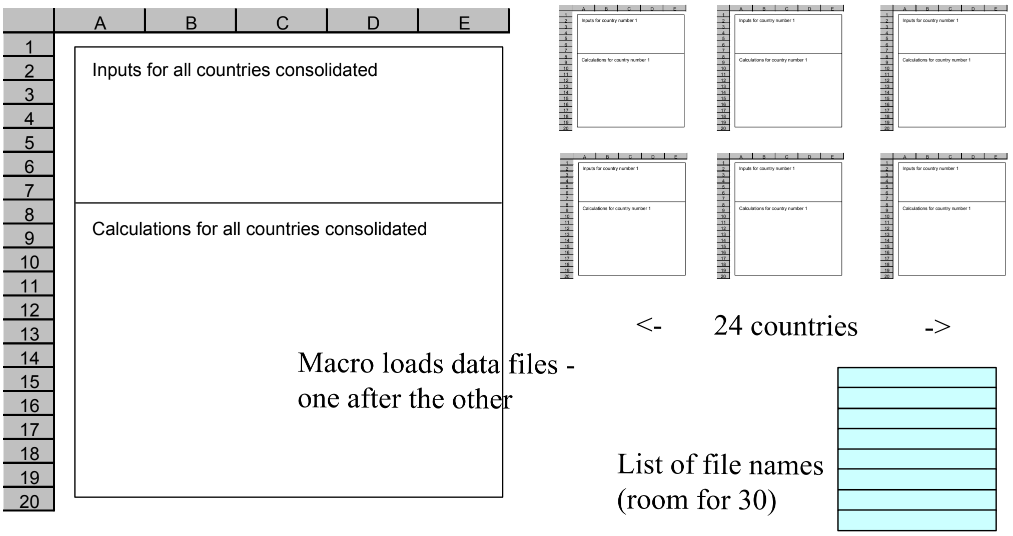

A
B
C
D
E
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Inputs for all countries consolidated
Calculations for all countries consolidated
Macro loads data files - one after the other
Inputs for country number 1
Calculations for country number 1
<- 24 countries ->
List of file names (room for 30)

Figure 31: Consolidation using macros

The advantages of this method are that:

- the consolidation model is small and will run quickly;
- if the macro is well written, the model is easy to use; and
- by leaving room for expansion on the list of files, adding an extra country is easy.

Disadvantages are that:

- writing the model will require a detailed understanding of programming macros; and
- the calculation method is not transparent, so you have to be confident that the macro contains no errors.

Macros

Macros can be used to automate almost any operation in a spreadsheet. Macros are in no way an essential part of best practice model design, but they can be useful. Macros can easily be linked to buttons placed on the spreadsheet, toolbar buttons or menu commands. There are also pitfalls to avoid when using macros.

Macros can be used in a spreadsheet in two basic ways.

Macros fundamental to a spreadsheet design

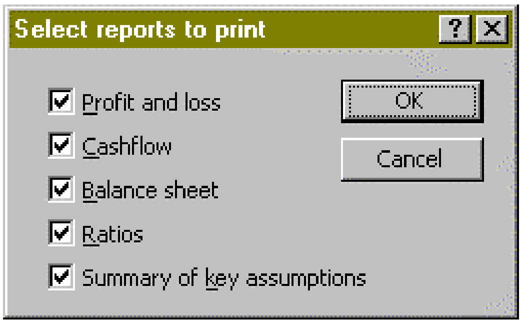
Macros can be used as part of a spreadsheet design, for example to import or manipulate the data used by the model. This can be a useful way to automate a routine piece of work that would otherwise be time consuming and prone to errors. However, writing macros which copy blocks of data about is itself a risky process.

When a macro is essential to the working of a model, or is large and complicated, you should plan how the macro will work. The simplicity of recording basic macros makes it tempting to skip this stage.

Macros to make a spreadsheet easier to use

Macros can also be used as an addition to a model design, to make the spreadsheet easier to use. Such macros can often be very simple. For example:

- a print macro, linked to a button at the top of each results sheet, makes printing commonly used reports much quicker;
- dialog boxes for key operations can make a model more user friendly and give a professional finish; and
- a navigation macro can make it easier for users to find their way around a large model.



☑ Profit and loss
☑ Cashflow
☑ Balance sheet
☑ Ratios
☑ Summary of key assumptions

Figure 32: Example dialog box

Chapter summary **Design**

- Before you design a spreadsheet, make sure that a spreadsheet is the most appropriate piece of software for the job.

- The six golden rules of spreadsheet design can help you write models that are easier to understand or update and are less likely to contain errors.

1. Always separate inputs from calculations and results.

2. Use only one unique formula in each row or column of each worksheet.

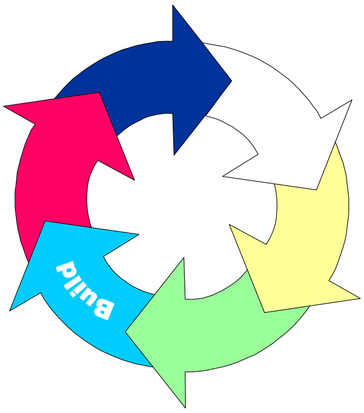
3. A well designed spreadsheet can be read like a book: from front to back, top to bottom and left to right.

4. Use multiple worksheets, so you can insert rows and columns into your work.

5. Use each column for the same purpose throughout the model.

6. Include a documentation sheet so that the model user knows who developed the model and which version they are using.

6 Build



In the Build stage, the coding of the model is undertaken. This stage is easier, quicker and less prone to errors if the specification and design stages are successfully completed.

It is always tempting to start coding the model too soon, especially when you are under pressure to produce results from the model quickly. Taking time to understand the problem and how you are going to solve it makes building the model:

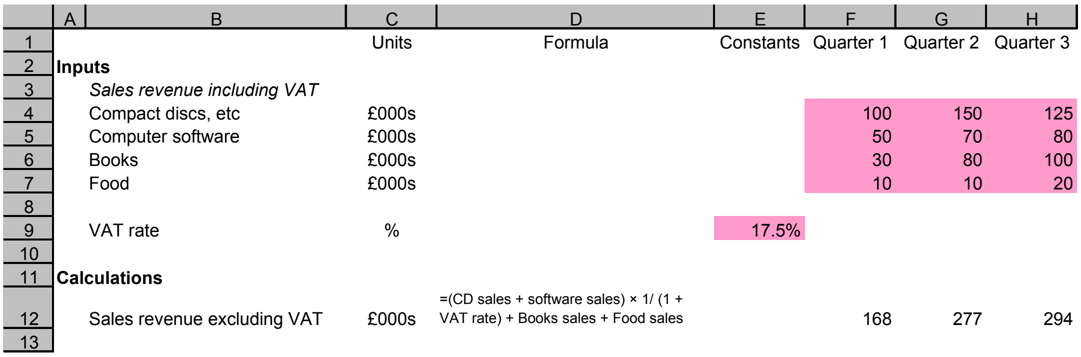
- quicker and easier, because you have a model specification that describes what the model will do rather than having to work it out as you go along;
- less prone to errors if you have a written description of how the model works; and
- less likely to have to be reworked, if you have taken some time to build a common understanding of the requirement of the model.

Hints and tips for model building

The following are all suggestions to bear in mind while building your model. They have a similar aim to the hints and tips for model design: to make your finished model easier to understand and update and less prone to errors.

Keep formulae simple

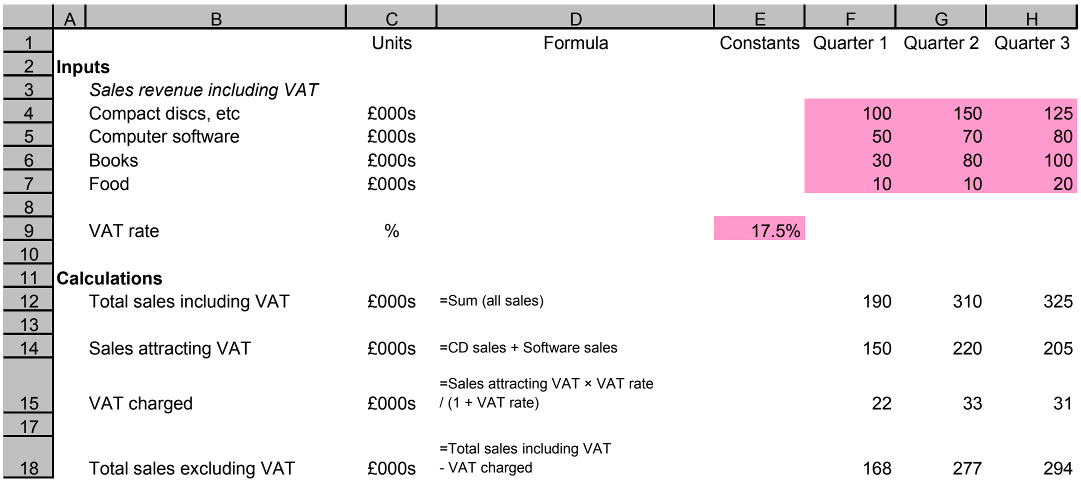
To make your spreadsheet as easy to understand as possible, keep the formulae simple. Just as a clearer writing style can be achieved by breaking up long sentences, a clearer model can be achieved by breaking up long formulae.



	A	B	C	D	E	F	G	H
1			Units	Formula	Constants	Quarter 1	Quarter 2	Quarter 3
2	**Inputs**							
3		*Sales revenue including VAT*						
4		Compact discs, etc	£000s			100	150	125
5		Computer software	£000s			50	70	80
6		Books	£000s			30	80	100
7		Food	£000s			10	10	20
8								
9		VAT rate	%		17.5%			
10								
11	**Calculations**							
12		Sales revenue excluding VAT	£000s	=(CD sales + software sales) × 1/ (1 + VAT rate) + Books sales + Food sales		168	277	294
13								

Figure 33: Example of complicated formula

The example above calculates sales revenue after excluding VAT. Although the formula is not exceptionally long, a first time reader would probably need to read it more than once before understanding what the formula is trying to do. Note that the calculation is assuming that books and food do not attract VAT, but does not make this assumption particularly clear.



	A	B	C	D	E	F	G	H
1			Units	Formula	Constants	Quarter 1	Quarter 2	Quarter 3
2	**Inputs**							
3		*Sales revenue including VAT*						
4		Compact discs, etc	£000s			100	150	125
5		Computer software	£000s			50	70	80
6		Books	£000s			30	80	100
7		Food	£000s			10	10	20
8								
9		VAT rate	%		17.5%			
10								
11	**Calculations**							
12		Total sales including VAT	£000s	=Sum (all sales)		190	310	325
13								
14		Sales attracting VAT	£000s	=CD sales + Software sales		150	220	205
15		VAT charged	£000s	=Sales attracting VAT × VAT rate / (1 + VAT rate)		22	33	31
17								
18		Total sales excluding VAT	£000s	=Total sales including VAT - VAT charged		168	277	294

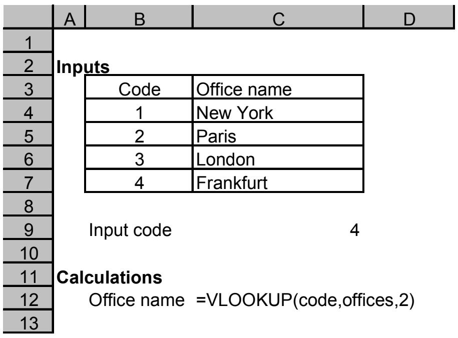
Figure 34: Example of simpler formulae

This second example is slightly longer but performs the same calculation. But it does so in a series of much simpler steps. By breaking your calculations into a series of steps you make them easier to understand and reduce the chances of making an error. In most models, the intermediate steps introduced here are worth including for use in other calculations.

One of the common ways that formulae become too long is by including multiple IF commands. Frequently, this problem can be overcome by using lookup functions. For example, consider this formula for converting an input code number into an office location.

=IF (code=1, “New York”, IF (code=2, “Paris”, IF (code=3, “London”, IF (code=4, “Frankfurt”))))

Although the concept behind this formula is exceptionally simple, the resulting formula is rather awkward. Consider instead the example below.



	A	B	C	D
1				
2	**Inputs**			
3		Code	Office name	
4		1	New York	
5		2	Paris	
6		3	London	
7		4	Frankfurt	
8				
9		Input code	4	
10				
11	**Calculations**			
12		Office name	=VLOOKUP(code,offices,2)	
13				

Figure 35: Example of lookup function

This example uses the VLOOKUP function to calculate the answer with a much shorter formula. As well as being easier to understand, it is also much easier to update, were the company to acquire an additional office.

Note also the use of range names in this example. The name ‘code’ has been given to cell C9. The name ‘offices’ has been given to the range B3:C7. As a result, the formula is much easier to read and understand.

Use of named cells and ranges

Allocating meaningful range names to areas or cells within a spreadsheet can speed up the development process, make the model easier to understand and reduce the risk of errors made by referring to the wrong cell.

Using range names you can write more meaningful formulae, for example:

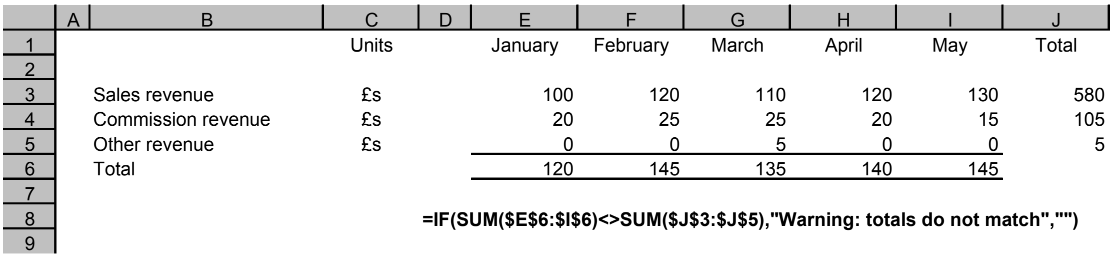
Operating profit	= Total_sales – Total_costs
Total sales	= SUM (Sales)
Units sold (in cell G10)	= G9 * (1 + growth_rate)

When referring to nearby cells, most people find a formula based on cell references, such as =A8 – A9, easier to follow than using names, such as = Total_sales – Total_costs. But when a cell is referred to in a formula a long way away from where it is calculated, range names are very useful. For example, if your model contains a number of net present value calculations, a named cell "discount_rate" will make your calculations much clearer.

Build in error traps

It is good practice to include checks for errors in a model. These can be set up to check for errors in the input data provided, or for errors in the model formulae.

Cross casting is the process of checking that the totals in the rows and columns of a spreadsheet are consistent. Consider the example below.



	A	B	C	D	E	F	G	H	I	J
1			Units		January	February	March	April	May	Total
2										
3		Sales revenue	£s		100	120	110	120	130	580
4		Commission revenue	£s		20	25	25	20	15	105
5		Other revenue	£s		0	0	5	0	0	5
6		Total			120	145	135	140	145	
7										
8				=IF(SUM($E$6:$I$6)<>SUM($J$3:$J$5),"Warning: totals do not match","")						
9										

Figure 36: Example of cross casting

In this example, an error message formula is being placed in cell D8 to check that the row and column totals match. It will prove useful because the total in row 6 has missed out the 'Other revenue' line, affecting the total for March.

When functions more complex than simple addition are used, this method can cause problems because Excel stores results to a greater degree of accuracy than its calculations actually achieve. It is safer to use a formula of the form:

=IF (ROUND (SUM($E$6:$I$6) – SUM($J$3:$J$5), 1) < > 0, "Warning: totals do not match", "")

The ROUND function will ensure that the error message only appears if an error more significant than one decimal place has occurred.

Error traps can be usefully included:

- on balancing financial statements, such as a balance sheet or a sources and applications of funds statement;

- when equivalent data is provided from more than one source, for example when a total revenue is both input and calculated from individual revenue lines; and

- when it is possible to perform a calculation in two equivalent but different ways.

When you use error traps, make sure you put them somewhere noticeable. For example, try placing references to all of your error traps in a single block at the top of one of the front sheets.

Built in features

When a model is going to be used over a long period of time, especially if it is to be used by a large number of people, Excel 97 provides a number of facilities to make your spreadsheet more robust. For example, you can:

- use cell protection to protect formulae in the model from being overtyped, while still allowing input values to be changed;

- use the data validation feature, such that an error message will appear if inputs lie outside a valid range.

Model building by more than one person

The Build stage of the modelling life cycle is a job best suited to one person alone. It is only possible for one person to edit a master copy of a spreadsheet at any given time. And, although good design and co-operation will reduce differences in approach, two modellers will always have slight differences in style and preferences.

However, when time for development is at a premium, it may be necessary to double up the building resource. If forced to do so, you will probably find it helpful to:

- agree the boundaries and connections between the parts of the model being developed by different people. A modular design can help here too: if all of the cells containing connections to a different part of the model are placed in the same part of the sheet, it is easier to connect the sections together;

- always know who is editing the master copy of the model, and who is making peripheral changes which will be incorporated to the master copy later;

- inserting an extra row or column can cause havoc if someone else is still working on a separate section of the model. Try to only insert rows when all changes have been consolidated;

- take care when copying different sections of the model together that you remove any external links to other model versions. It is easy to find external links by choosing ‘Links…’ from the Edit menu;

- make sure that everyone uses exactly the same conventions for style of text, colour coding, and so on; and

- maintain close communication between the model builders all the time.

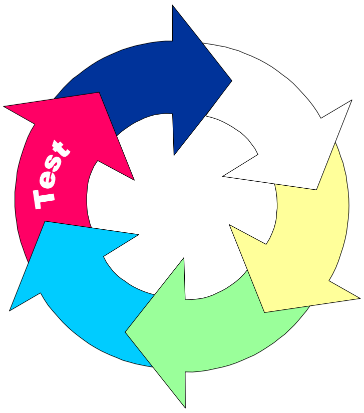
Chapter summary **Use**

- The Build stage is quicker, easier and less prone to errors if the specify and design stages have been successfully completed.

- Make calculations easy to understand by breaking up long formulae into simpler steps.

- Reduce the risk of errors by including error traps in the model code.

7 Test



Why test?

A best practice spreadsheet model can be relied upon for important decisions. This is only possible if you have confidence that the results produced by the model are reliable. It is not possible to guarantee that even a moderately complex model is error free. Testing can, however, substantially reduce the risk of significant errors in the model.

If testing is skipped or done poorly, errors are likely to be discovered after the model has been put into use. Errors at this stage can undermine the credibility of both the model and its developer. The value of testing can be measured against the potential cost of a wrong decision: if a model is being used for an important and expensive business decision, the time and resource spent testing the model is time and money well spent.

Who should test?

The objective of testing a model is to demonstrate that the model does not work as intended, not that it does work correctly. It is impossible for the model builder to be sufficiently critical of his or her own work. As a result, testing should always be carried out by an independent third party.

It can be appropriate for testing to be done by a member of the team who has been involved with other aspects of the overall project, or aspects of the modelling other than the build stage – there are obviously benefits of using someone who is familiar with some of the issues that the model reflects. However, the technical skills required for a good tester are similar to those required for building the model. The tester should be able to understand and critique the detail of the spreadsheet code.

When to test?

When there is plenty of time between the development of a model and the need for results, testing can take place between the build and use stages of the modelling life cycle. Under more pressing circumstances, it can be more difficult to identify the most appropriate time to test. For example:

- when there is great pressure for immediate results from the model there is a trade off to be made between gaining confidence in the model's results and trying to produce some reasonable answers quickly;
- when the model develops in a number of incremental stages it is difficult to know when the time is right to first test the model, or when further changes are significant enough to require retesting.

There is no single correct answer to these questions, but there are key points in time when testing should be considered. For example, test the model:

- when the model is complete, or at least when no further changes are envisaged to the logic in the model, and the model will continue to be used;

- before model results are to be used for key decisions, allowing sufficient time for the testing to be carried out and the required corrections to be made to the model before the deadline arrives;

- when enough changes have been made to the model since the previous test to make material changes to the results possible. You need to consider how big a change in the results constitutes a material change.

Types of test

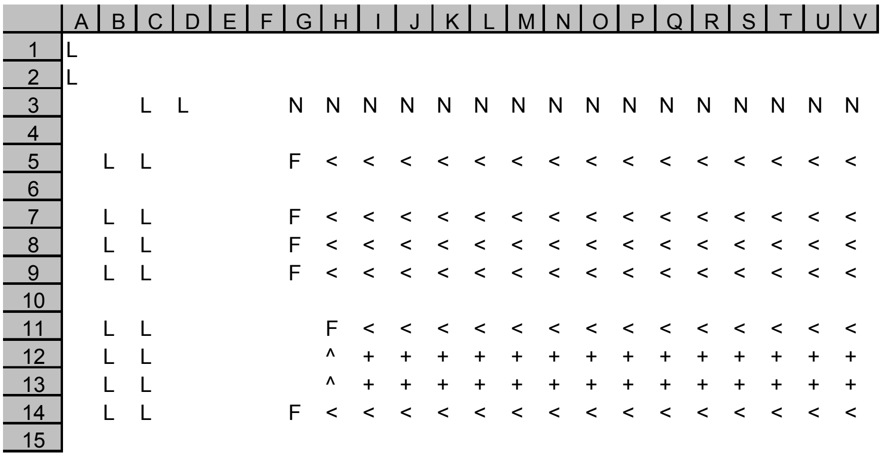
Specification test

One of the objectives of the specification stage was to establish a common understanding of how the model should work. To achieve this, it is necessary to test that the specification agrees with the model sponsor's understanding of what the model will do. Clearly, this does not require the model build to have been completed, or even started.

Unique formulae test

To test that the formulae in the model have been written correctly, you can check the model itself against the logical assumptions described in the model specification, or similar document. A review of every unique formula in the model is an essential part of any thorough model test. But the number of formulae in a reasonably large model makes this a daunting task.

We recommend resolving this problem by using the formula maps produced by a spreadsheet auditor package, such as the one below produced by Spreadsheet Professional. We discuss the use of spreadsheet auditing software in Appendix A.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	L																					
2	L																					
3			L	L			N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
4																						
5		L	L				F	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
6																						
7		L	L				F	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
8		L	L				F	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
9		L	L				F	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
10																						
11		L	L					F	<	<	<	<	<	<	<	<	<	<	<	<	<	<
12		L	L					^	+	+	+	+	+	+	+	+	+	+	+	+	+	+
13		L	L					^	+	+	+	+	+	+	+	+	+	+	+	+	+	+
14		L	L				F	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
15																						

Figure 37: Example formula map

The map identifies the content of each cell. An ‘L’ indicates a cell that contains a text label, an ‘N’ a cell that contains a numeric input while the other symbols represent cells that contain formulae. An ‘F’ indicates a unique formula while the other symbols represent formula that have been copied, either from the row above or the column to the left.

Code	Meaning
L	Text label
N	Numeric input
F	Formula
<	Formula that is a copy of the cell to the left
^	Formula that is a copy of the cell above
+	Formula that is a copy of both the cell to the left and the cell above

Figure 38: Map legend

Notice that the example map is of a spreadsheet that follows the second golden rule of spreadsheet design: it uses just one unique formula for each row. As we said in the design chapter, following this rule makes for effective testing because any rogue F in the middle of a row of this spreadsheet map will immediately be noticed by the tester. In this example, the tester needs to check only the formulae in the leading column. The substantial job of testing all 125 formulae in this spreadsheet has been reduced to the more reasonable one of testing just 6 unique formulae.

To get the most out of the formula maps, we recommend that you use them throughout the formula testing. For each F on the map, mark it with a tick to show that it is correct or a description if not, together with any other relevant comments. Together with a print out of the corresponding model sheets, this forms a useful record of the testing process.

Numeric tests

While the unique formula test concentrates on the logic within the model, whatever inputs are provided, there are a number of different tests to increase your confidence in the validity of the numbers currently being produced by your model.

There can be various ways to subject the results from your model to independent verification. For example:

- with appropriate simplifying assumptions, it may be possible to reproduce some of the model results by hand, or in a “quick and dirty” independent model;

- when the model is developed to replace a precursor model any similarities or differences in common results can be explained by differing assumptions; and

- when you know that there will not be time to test the model, you could get two modellers to independently do the analysis and then compare their results.

A well designed model will contain some internal numerical tests. A balancing balance sheet and consistency between calculated totals and sub-totals are both necessary signs of a reliable model, but are most certainly not proof that there are no errors still in the model.

You will only get sensible results out of a model if you put sensible input assumptions in. While it is often outside the scope of the independent tester's work, the validity of the input assumptions will obviously affect the appropriateness of the answers.

Robustness tests

The longer the expected life span of the model and the more different users of the model there will be the more difficult the model test becomes. Instead of testing the validity of the model just under current circumstances, the tester needs to consider all of the feasible circumstances under which the model may be used in future. Below are some different ways to test the robustness of a model.

An environment test is a test of the operation of the model on the computer or computers where the model will reside. Points to check are the hardware specification of the computer to be used, network and printer operations and the particular versions of software being used.

Boundary tests are on the behaviour of the model when inputs reach sensible limits. For example:

- how does the model cope with very large or very small inputs?
- does typing in a zero produce division by zero errors?

If you type junk into a model, you expect to get junk out, although for many models some form of data validation may be appropriate. But extreme cases of using particularly good or bad input assumptions are often important sets of model results, so the model should be able to cope with these.

Macro tests

If the model contains macros, testing these can be particularly time consuming, because they can sometimes cause errors only in particularly unusual circumstances. Things to check include:

- do the buttons, dialog boxes and menus work as appropriate?
- how does the macro cope with inappropriate data? how elegantly does it crash?
- if the macro involves file handling, what happens with duplicate or non-existent files?
- does the macro assume any default settings for the spreadsheet, such as sheet names, which a user may change?
- what happens when the user presses Escape during the macro execution?

Type of test	**Questions to ask**
Specification test	Does the model specification make the agreed logical assumptions about the business problem?
Formulae test	Has the model specification been appropriately coded in the model? Have formulae been copied appropriately across rows or down columns?
Numeric tests	Can you reproduce some part of the model results independently from the model itself? Is it possible to reconcile the results with another source, such as a precursor model? What are the most surprising conclusions of the model results?
Robustness tests	Does the model work on all of the computers where it will be used? How does the model cope with a wide range of differing input values?
Macro tests	Can the macros cope with all of the situations that might arise?

Figure 39: Summary of types of tests

Model types

Some form of testing is appropriate for all types of model. However, the types of tests and time required will differ for the types of modelling project that we first introduced on page 2-8.

Model type	Features of the test stage
Complex models	• Using an independent tester is the most rigorous way to test a model • As a rule of thumb, the build and test stages should take about the same amount of time
Simple models	• When you are developing a model for personal use, it can be difficult to find an appropriate independent tester • If you are forced to test your own model, concentrate on verifying results against alternative sources because rigorously checking the code of your own model is extremely difficult and it is easy to miss things
Time-critical	• If you have to meet a series of tight deadlines, it is difficult to know the best time to test the model • If you produce results from an untested model, always label the model output as untested • If you plan to make decisions based on an untested model, make sure that you understand the implications on your decision of likely errors • If you know that there will not be time to test the model, you could ask two modellers to independently produce the same piece of analysis
Ill-defined	• It is difficult to know when to test the model if assumptions are regularly changing • If you have to produce results based on an inadequately tested model, make sure that they are clearly labelled as such in a header or footer

Figure 40: Methods for testing different model types

Test plan

A test plan can be useful to provide a structure to the testing process and ensure that you miss nothing out. It is particularly helpful to help split responsibilities when more than one person is involved in the testing or to give additional guidance to an inexperienced tester.

The sample test plan below is suitable for a model that is fairly simple, except that it contains some significant macros. The plan will vary for the particular model being tested.

Type of test	**Tasks**
Formulae test	Produce formula maps for every sheet in the model. Check each unique formula in the model. Check for the common mistakes in copying formulae. Identify the most difficult formulae in the model and check their syntax.
Numeric tests	Reconcile the results for the first three years against those produced by the budget planning model. Calculate net profit for the second year of the model from the input data without making use of any of the model calculations. Identify the three most significant conclusions of the model results and trace their calculation back through the model.
Robustness tests	Perform boundary tests with a range of sensible input values.
Macro tests	Check the data consolidation macro, including: • with an error in the input file; • with the input file missing; and • on the user's computer with files from their network connections. Check that the print macros all work.

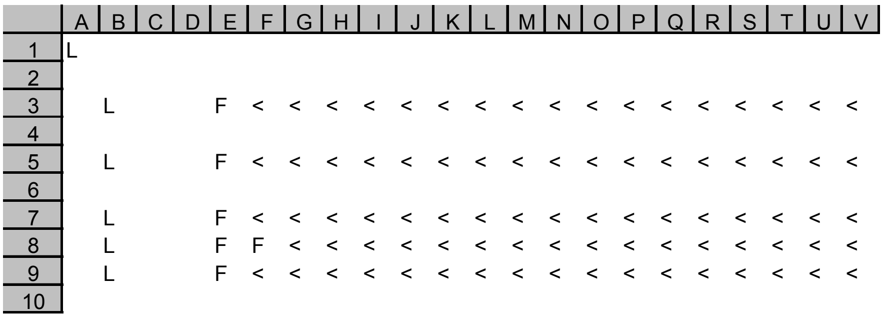
Figure 41: Sample test plan

Common errors

With experience, you can learn to spot many of the most common spreadsheet errors very quickly, particularly if you use formula maps as part of your test plan. For a quick and thorough test, it pays to be alert for the most common types of errors, which we discuss below.

Formulae not copied

In the example below, the formulae in row 8 stand out. One of the easiest ways to introduce an error into a model is to update the formula in a cell (in this example in cell E8) and forget to copy the new formula across into the other cells in the row. By using a spreadsheet map and a design in which all formulae are copied across all rows, this mistake can be noticed very quickly.



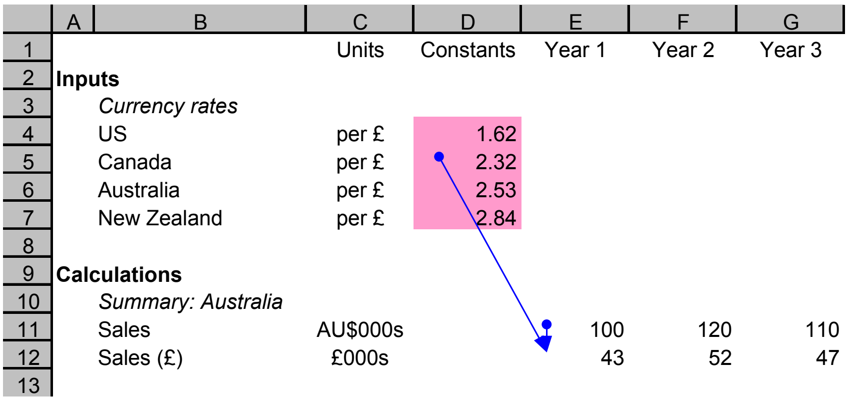
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	L																					
2																						
3		L			F	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
4																						
5		L			F	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
6																						
7		L			F	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
8		L			F	F	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
9		L			F	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<
10																						

Figure 42: Example formula map: formula not copied across

Wrong reference

Nearly every formula in a spreadsheet refers back to another input or calculation. With the quantity of references in any large spreadsheet, it is inevitable that you will make mistakes and refer to the wrong cell. Sometimes, the resulting formula will produce a meaningless result making it easy to spot with some simple numerical testing. If you are unlucky, the error in the result will be more subtle.

The only way to check for wrong references with any confidence is to go through a formula map, checking every unique formula. In Excel, checking references is made easier if you use the auditing toolbar. This allows you to trace the precedent cells graphically, as in Figure 43 below.



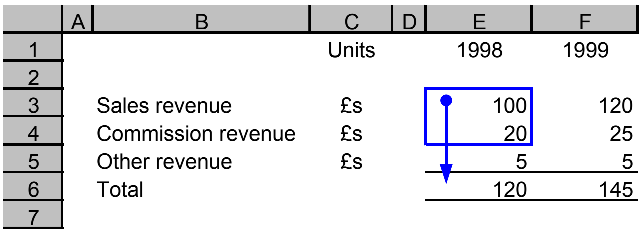
	A	B	C	D	E	F	G
1			Units	Constants	Year 1	Year 2	Year 3
2	**Inputs**						
3		*Currency rates*					
4		US	per £	1.62			
5		Canada	per £	2.32			
6		Australia	per £	2.53			
7		New Zealand	per £	2.84			
8							
9	**Calculations**						
10		*Summary: Australia*					
11		Sales	AU$000s		100	120	110
12		Sales (£)	£000s		43	52	47
13							

Figure 43: Example formula map - wrong reference

Sum over the wrong range

A similar mistake is to include the wrong cell reference in a SUM formula. It is particularly easy to introduce this error when you insert an extra row in a block of cells that are being summed. Insert a row in the middle of the block, and the formula will automatically adjust to include the extra row, but insert a row immediately above or below the block and the new row will be omitted from the formula.

You can build internal checks of the sums in a spreadsheet by using cross casting, as described in the build chapter. When testing a model, Excel's trace precedents feature helps to find errors of this type by showing them graphically, as in Figure 44 below.

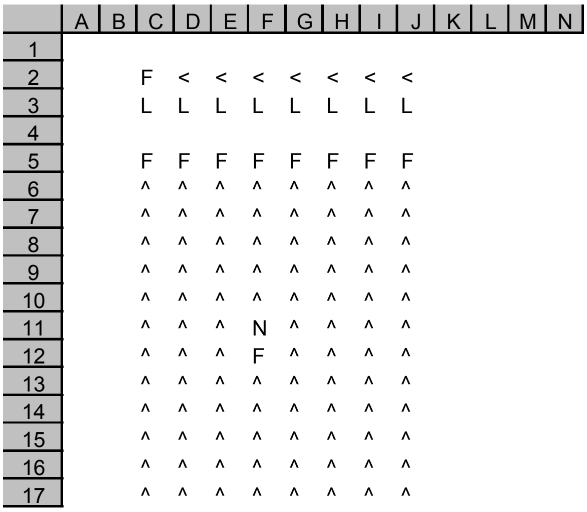


	A	B	C	D	E	F
1			Units		1998	1999
2						
3		Sales revenue	£s		100	120
4		Commission revenue	£s		20	25
5		Other revenue	£s		5	5
6		Total			120	145
7						

Figure 44: Example of a sum over the wrong range

Temporary fix

In Figure 45, most columns have a single unique formula. However, someone has introduced a rogue input into column F. A stray N on a formula map is usually caused by a temporary 'fix' being introduced to overcome some problem. In this example, it is possible that the change is intentional (although not best practice) – but it is all too easy to forget a fix of this sort and leave it to cause errors later on, when the data in the model has been changed.



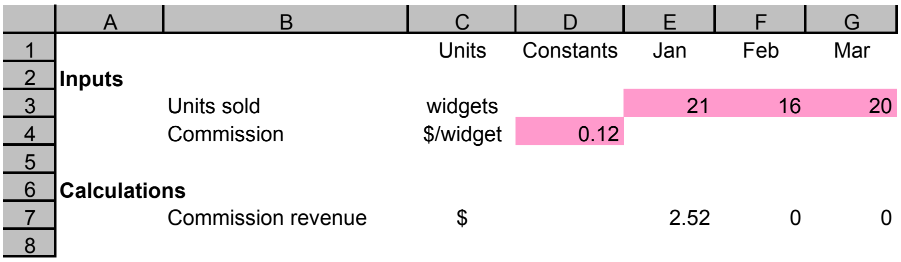
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2			F	<	<	<	<	<	<	<				
3			L	L	L	L	L	L	L	L				
4														
5			F	F	F	F	F	F	F	F				
6			^	^	^	^	^	^	^	^				
7			^	^	^	^	^	^	^	^				
8			^	^	^	^	^	^	^	^				
9			^	^	^	^	^	^	^	^				
10			^	^	^	^	^	^	^	^				
11			^	^	^	N	^	^	^	^				
12			^	^	^	F	^	^	^	^				
13			^	^	^	^	^	^	^	^				
14			^	^	^	^	^	^	^	^				
15			^	^	^	^	^	^	^	^				
16			^	^	^	^	^	^	^	^				
17			^	^	^	^	^	^	^	^				

Figure 45: Example formula map - temporary fix

Relative and absolute references

Another commonly found error is caused by confusion between relative and absolute references. Remember that a cell reference in a formula of the form '=D4' will change if you copy the formula across the row to E4, F4 and so on. Copied down a column it will change to D5, D6, etc. If you use the reference '=$D$4' it will not change when copied across or down. You can also use semi-absolute references of the form $D4 or D$4.

The most common mistake is probably to use a relative reference instead of an absolute one, such as in the example below. This is quite easy to spot numerically, but if you use an absolute reference in place of a relative one it can be much more difficult to spot. Again, the only way to find this mistake reliably is to use a formula map to go through all of the unique formulae in the spreadsheet.



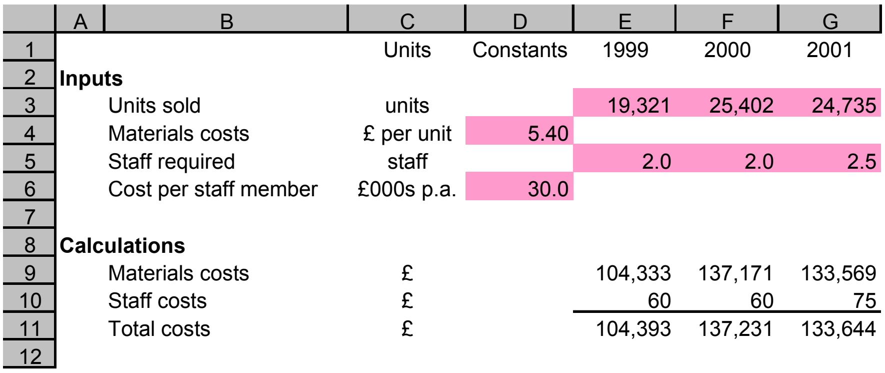
	A	B	C	D	E	F	G
1			Units	Constants	Jan	Feb	Mar
2	Inputs						
3		Units sold	widgets		21	16	20
4		Commission	$/widget	0.12			
5							
6	Calculations						
7		Commission revenue	$		2.52	0	0
8							

Figure 46: Example of absolute / relative reference error

One technique to help avoid making this mistake is by following the fifth golden rule of spreadsheet design, by using each column for the same purpose throughout the spreadsheet. If you always use column D for constants, you know that every formula that refers to column D should contain an absolute reference, such as $D$4, while references to others columns should always be relative, for example E3.

Units errors

Mixing up the appropriate units for the elements in a calculation is another frequently occurring problem. Especially common is confusion between the order of magnitude for inputs, such as £s versus £000s.



	A	B	C	D	E	F	G
1			Units	Constants	1999	2000	2001
2	Inputs						
3		Units sold	units		19,321	25,402	24,735
4		Materials costs	£ per unit	5.40			
5		Staff required	staff		2.0	2.0	2.5
6		Cost per staff member	£000s p.a.	30.0			
7							
8	Calculations						
9		Materials costs	£		104,333	137,171	133,569
10		Staff costs	£		60	60	75
11		Total costs	£		104,393	137,231	133,644
12							

Figure 47: Example of units error

Note that the error in Figure 47 is particularly easy to identify because of two features of the design. This example spreadsheet:

- includes a units column to make tracing through the calculations clearer; and
- splits the calculation of total costs into its intermediate steps, so that the odd result for staff costs is much easier to spot.

It is also good practice to try to avoid switching between units, for example from £s to £000s, except when absolutely necessary.

Commonly misused functions

Certain functions are frequently used incorrectly. In financial models, the most commonly misunderstood is the NPV function, used to calculate net present values.

The NPV function takes the form =NPV (rate, value1, value2, …) where rate is an appropriate discount rate and value1, value2, etc. are a stream of cash flows to be discounted. One common error with the NPV function is using an inappropriate discount rate, especially when the model contains calculations in a mixture of real and nominal prices.

A second common mistake concerns the timing of cash flows. By default, the NPV function assumes that all cash flows occur at the end of the time period you are considering. For an ongoing business it is usually more appropriate to assume that cash flows occur in the middle of each time period. For a model based on annual time periods, using a discount rate of 10% and an end year rather than a mid year assumption decreases the present value calculated by 5%.

You can work around these problems by making an appropriate adjustment to the NPV function. Alternatively, it is often easier to calculate present values from first principle or use the Excel add-in function XNPV, which allows you to explicitly state the dates on which cash flows occur.

Other functions that often cause errors are:

- lookup and reference functions, such as VLOOKUP, HLOOKUP, INDEX and MATCH; and
- complicated IF statements, especially when the model developer creates over complex formulae by nesting a series of IF statements.

These functions are often very useful, but make sure that you understand exactly how they work before using them in your model. By making the formulae in your model easy to understand, you reduce the risk of introducing errors and increase the chances that a tester will find your mistakes.

The test file

Testing is concerned with increasing confidence in the reliability of the model's results, so it is important that the testing process itself inspires confidence. A well documented testing process does this. A test file is a useful document to illustrate how the model was tested and a demonstration that reasonable steps were taken to establish the accuracy of the model.

The test file should contain a record of the different tests that were carried out, annotated print outs of the model output checked (including the formula maps), an auditable trail of the errors found and corrections made and any other supporting documentation, such as the results of independent numerical tests.

Change requests

An important part of the test documentation is a record of the changes recommended to the model, and how they were implemented. A system of change request forms can be used to track model changes.

An example form is included on page 7-71. The form is split into three sections.

The tester completes the first section, which includes a description of the error found and, if appropriate, a suggested correction. The description should contain sufficient information for the model builder to correct most errors without having to spend time redoing the original test.

The model builder takes the completed forms and makes the required changes, completing the second part of the form. We do not recommend that the tester makes the changes to the model, because:

- the model builder usually understands the detail of the model better than anyone else. Change requests are sometimes the result of a misunderstanding by the tester rather than an actual error; and

- when more than one person is working on the model at one time, it is important that a single copy is kept as the master version. The model builder can continue to work with the master version while testing is underway.

The third part of the form should be completed by the tester to check that the change has been made appropriately. Retesting changes is important because changes to a model, incorrectly made, can have knock on effects on other parts of the model.

When going through the testing process, using a different version number for each batch of model changes will allow you to keep track of the effect of each correction. It will also make the test file clearer and the process easier to understand.

It can be useful to classify requests on the change request form depending on their importance. For example:

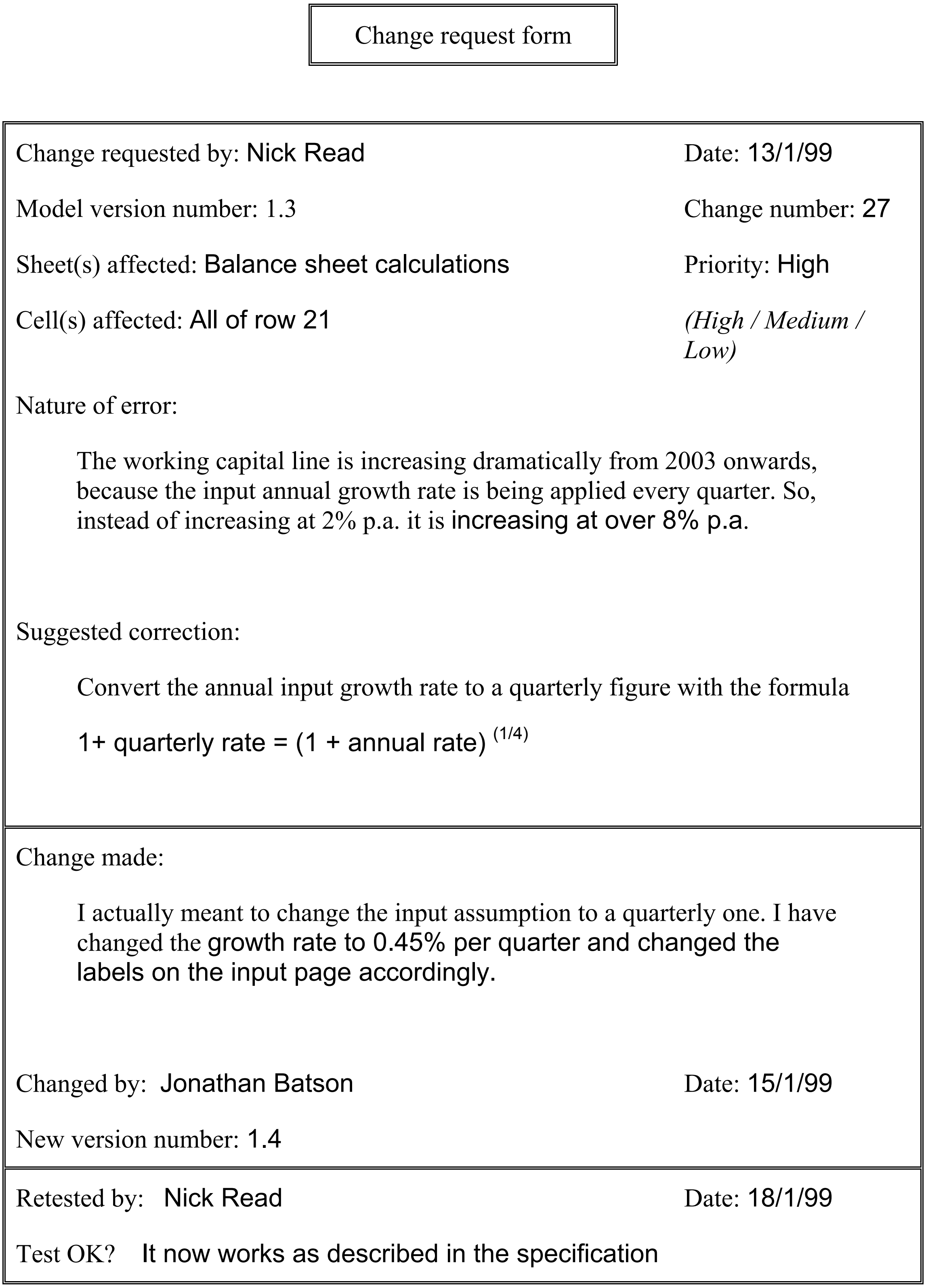
- high priority for errors which materially change the results produced by the model;

- medium priority for errors which have a very small effect or may affect the results in future; and

- low priority for changes which are recommended to make the model easier to understand, but will not affect the results quoted.

When time is limited, the builder can concentrate on high and medium priority changes and turn to low priority changes when time permits.

If you want to use change requests for your model testing, a blank sample form is provided in Appendix B to this guide, on page 10-96.



Change request form

Change requested by: Nick Read

Date: 13/1/99

Model version number: 1.3

Change number: 27

Sheet(s) affected: Balance sheet calculations

Priority: High

Cell(s) affected: All of row 21

(High / Medium / Low)

Nature of error:

The working capital line is increasing dramatically from 2003 onwards, because the input annual growth rate is being applied every quarter. So, instead of increasing at 2% p.a. it is increasing at over 8% p.a.

Suggested correction:

Convert the annual input growth rate to a quarterly figure with the formula

1+ quarterly rate = (1 + annual rate) $^{(1/4)}$

Change made:

I actually meant to change the input assumption to a quarterly one. I have changed the growth rate to 0.45% per quarter and changed the labels on the input page accordingly.

Changed by: Jonathan Batson

Date: 15/1/99

New version number: 1.4

Retested by: Nick Read

Date: 18/1/99

Test OK? It now works as described in the specification

Figure 48: Example change request form

Chapter summary **Test**

- Testing can reduce the risk of significant errors appearing in your model results and build up the credibility and influence of the finished model.

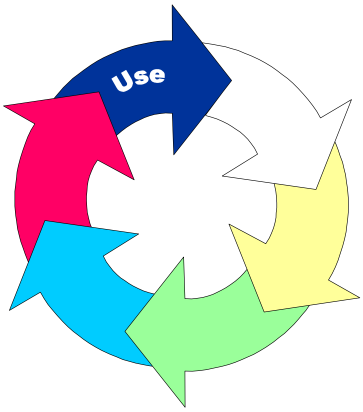
- Errors discovered after the model has been put into use undermine the credibility of the modeller.

- Testing should always be done by someone other than the model builder, someone who is motivated to find mistakes in the model.

- If the model builder uses only one unique formula per row or column, it is much easier to test all of the formulae in the model.

- Documenting the test process will help to build confidence in the finished model's results.

8 Use



Reaping the rewards of a best practice model

To use a model is not the same as to operate or run a model. Simply performing a model run and handing over a set of print outs is not good enough and risks wasting the benefits of developing a best practice model.

The model user usually understands the detail of how the model works better than anyone else. It is his or her responsibility to use this knowledge to understand, interpret and present the results from the model in the most useful way.

In this chapter, we will:

- discuss the best ways to present model results in tables and graphs;
- introduce some techniques for running and presenting alternative scenarios; and
- describe control techniques for keeping track of different model versions.

Presenting results

A big difference to the usefulness of a model can be made by effective presentation of results. Good presentation leads not just to easier communication of the results, but also improves the perception of the quality of the underlying model.

In general, try to:

- establish a standard look for all the reports from a model (or family of models), so that the reader immediately knows the source of any new reports. It also saves you time when developing additional reports;
- group lines into blocks of about five rows, so that the results can be understood in digestible blocks;
- use one, or at most two, typefaces. Achieve emphasis with sparing use of bold, italic or underlined text;
- draw boxes around cells to visually separate different results; and
- remember that shaded cells can become illegible if they are photocopied or faxed.

Headers and footers

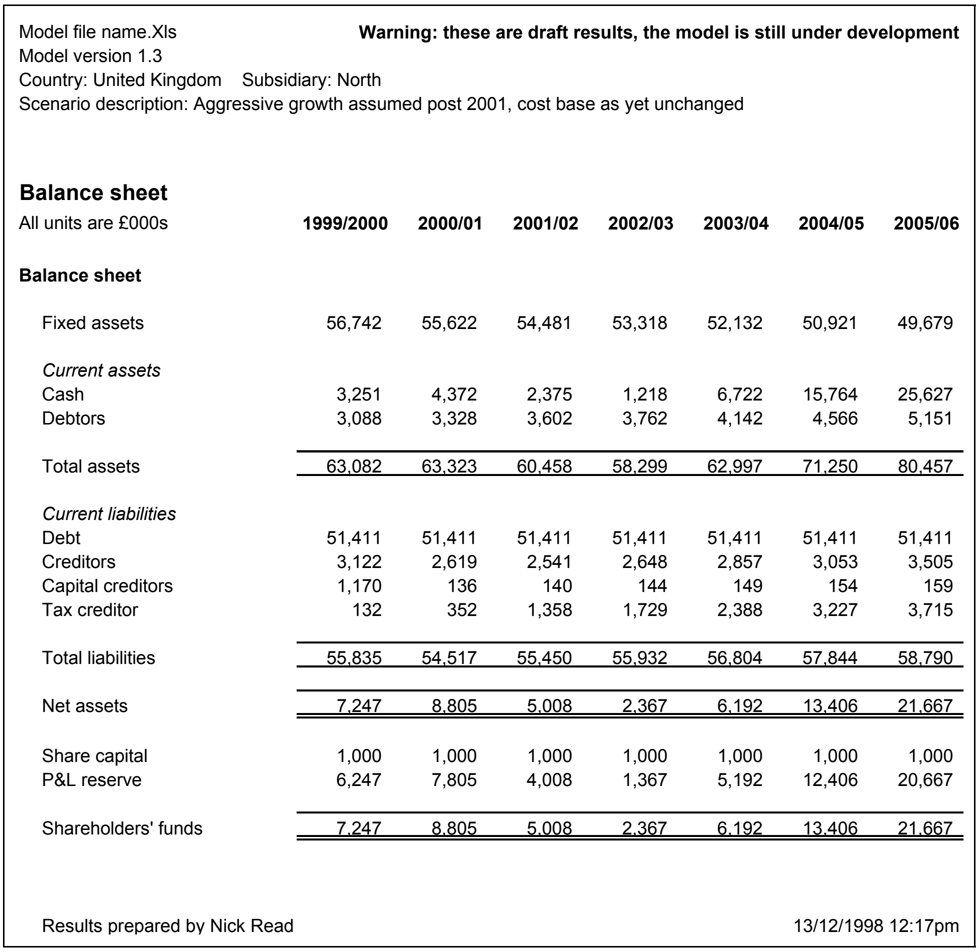
Putting key information onto report headers and footers makes it much easier to keep track of different model runs. Without them, it is easy to mix up out of date versions of model output with the correct version. Include on the header or footer:

- the model file name and/or the model name and version number;
- the date and time of printing;
- the model user's name;
- the data set used in the model; and
- a description of the scenario or purpose of the particular model run.

In addition, if you are using a model that has not yet been fully tested, make sure that you state this, for example with the header: "Draft results based on an untested model".

Excel 97 does not allow you to dynamically link a page header or footer to the contents of a cell, for example containing a description of the scenario being run. To resolve this problem, either:

- link each report in the model to a simple print macro that will set up the report header or footer based on the text in a particular cell; or
- place the a reference to the cell containing the scenario description in a cell on the report itself.



Model file name.Xls **Warning: these are draft results, the model is still under development**
Model version 1.3
Country: United Kingdom Subsidiary: North
Scenario description: Aggressive growth assumed post 2001, cost base as yet unchanged

Balance sheet

All units are £000s	**1999/2000**	**2000/01**	**2001/02**	**2002/03**	**2003/04**	**2004/05**	**2005/06**
Balance sheet							
Fixed assets	56,742	55,622	54,481	53,318	52,132	50,921	49,679
Current assets							
Cash	3,251	4,372	2,375	1,218	6,722	15,764	25,627
Debtors	3,088	3,328	3,602	3,762	4,142	4,566	5,151
Total assets	63,082	63,323	60,458	58,299	62,997	71,250	80,457
Current liabilities							
Debt	51,411	51,411	51,411	51,411	51,411	51,411	51,411
Creditors	3,122	2,619	2,541	2,648	2,857	3,053	3,505
Capital creditors	1,170	136	140	144	149	154	159
Tax creditor	132	352	1,358	1,729	2,388	3,227	3,715
Total liabilities	55,835	54,517	55,450	55,932	56,804	57,844	58,790
Net assets	7,247	8,805	5,008	2,367	6,192	13,406	21,667
Share capital	1,000	1,000	1,000	1,000	1,000	1,000	1,000
P&L reserve	6,247	7,805	4,008	1,367	5,192	12,406	20,667
Shareholders' funds	7,247	8,805	5,008	2,367	6,192	13,406	21,667

Results prepared by Nick Read 13/12/1998 12:17pm

Figure 49: Example report with header and footer

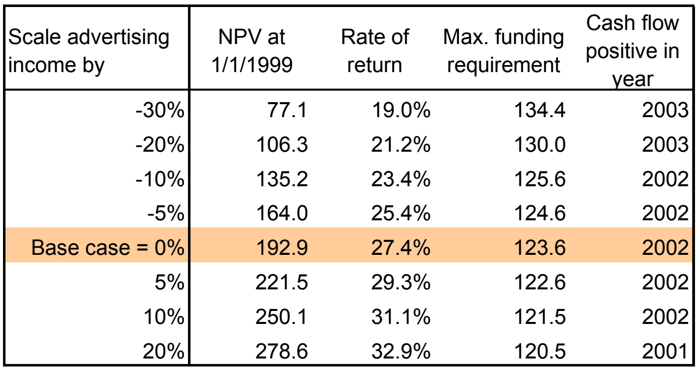
Techniques for running scenarios

Spreadsheet models are powerful tools for analysis. A successfully developed model can analyse a variety of alternative situations and inform the decision making process. Using a model to produce useful information requires skill, not just to generate computer printouts.

Data tables

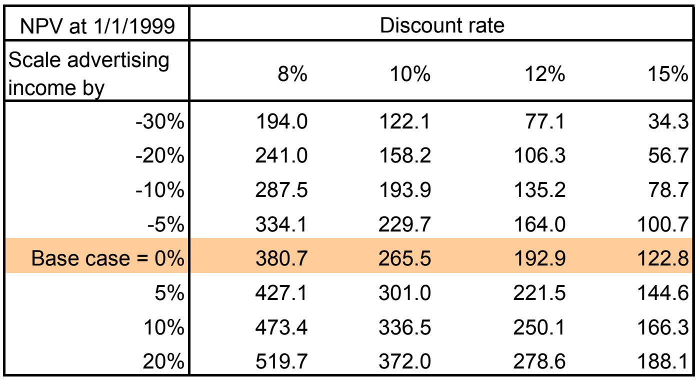
One of the most powerful ways that a model can be used to improve understanding of a problem is by illustrating the sensitivity of the results to varying some of the key input assumptions. It is very easy to understand how your input assumptions affect the results by using Data tables.

Data tables are quick and easy to produce and can be produced in one or two dimensions, to show the effect on the key results of varying one or two of your input assumptions. The example below illustrates how sensitive the overall performance of an imaginary, but realistic, project is to the level of advertising income assumed.



Scale advertising income by	NPV at 1/1/1999	Rate of return	Max. funding requirement	Cash flow positive in year
-30%	77.1	19.0%	134.4	2003
-20%	106.3	21.2%	130.0	2003
-10%	135.2	23.4%	125.6	2002
-5%	164.0	25.4%	124.6	2002
Base case = 0%	192.9	27.4%	123.6	2002
5%	221.5	29.3%	122.6	2002
10%	250.1	31.1%	121.5	2002
20%	278.6	32.9%	120.5	2001

Figure 50: Example of a one-dimensional data table



NPV at 1/1/1999	Discount rate			
Scale advertising income by	8%	10%	12%	15%
-30%	194.0	122.1	77.1	34.3
-20%	241.0	158.2	106.3	56.7
-10%	287.5	193.9	135.2	78.7
-5%	334.1	229.7	164.0	100.7
Base case = 0%	380.7	265.5	192.9	122.8
5%	427.1	301.0	221.5	144.6
10%	473.4	336.5	250.1	166.3
20%	519.7	372.0	278.6	188.1

Figure 51: Example of a two-dimensional data table

Reconciliation tables

Significant changes in model results can cause trouble for the model user. Even when the results have changed only because input assumptions have changed, it can reflect badly on the model user – unless you are able to provide a coherent description of why the results have changed. This is especially true if the changes in the results are caused by a series of smaller changes.

A good way to keep track of the effects of model changes is to keep an up to date reconciliation table, such as the one below.

Model	**Date**	**Project NPV (£m)**	**Max. funding requirement (£m)**
Original model	01/05	106.2	254.2
Revised revenue forecasts	03/05	101.1 *(-5.1)*	255.0 *(+0.8)*
Change to marketing costs	03/05	99.7 *(-1.4)*	265.1 *(+10.1)*
Correct error in interest calculation	04/05	98.9 *(-0.8)*	270.0 *(+4.9)*
Change to long term growth assumptions	08/05	90.2 *(-8.7)*	270.0 *(no change)*
Results of head count estimates	09/05	92.2 *(+2.0)*	261.6 *(-8.4)*

Figure 52: Example reconciliation table

When using reconciliation tables:

- try to separate the effect of changes to model logic, such as the change to the interest calculation above, from changes to the model input assumptions; and

- if the model logical assumptions are at all complicated, the effects of a series of changes are unlikely to be additive. So, the effect of reversing one of the changes in the list above might be different from the original change. However, the effect of the original change usually gives you a good idea of the likely result of reversing a change.

Armed with a reconciliation table, the model user can usually answer awkward questions about changes to model results. It allows people to focus on the significant causes of change, rather than worrying about the less important factors. It also helps to build confidence in the reliability of the model.

For an even quicker understanding of the significant effects, you can represent the reconciliation table graphically.

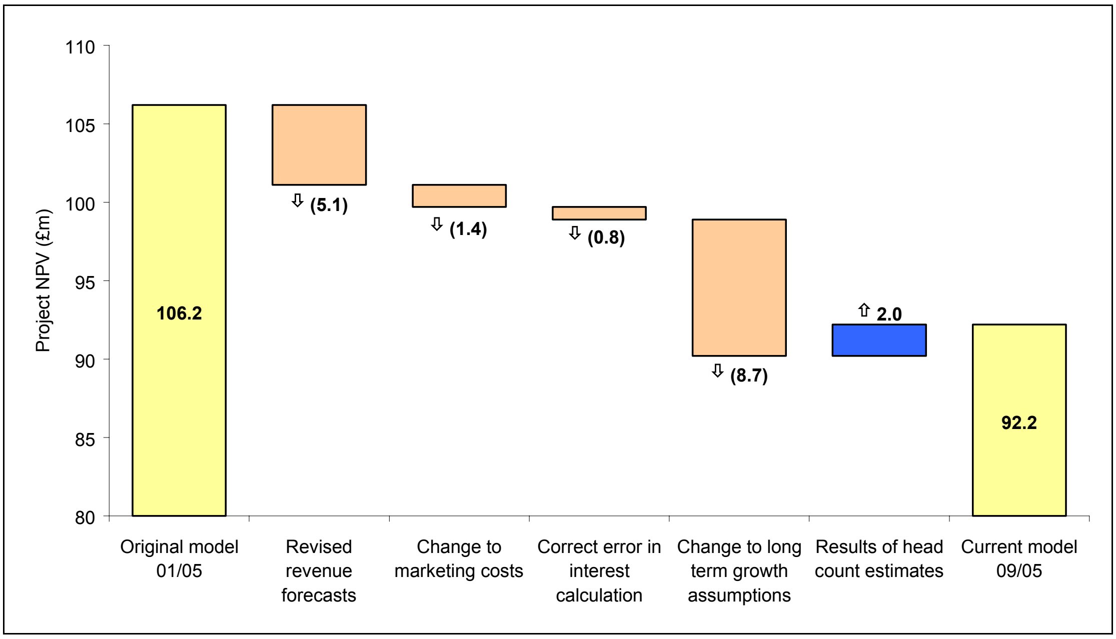
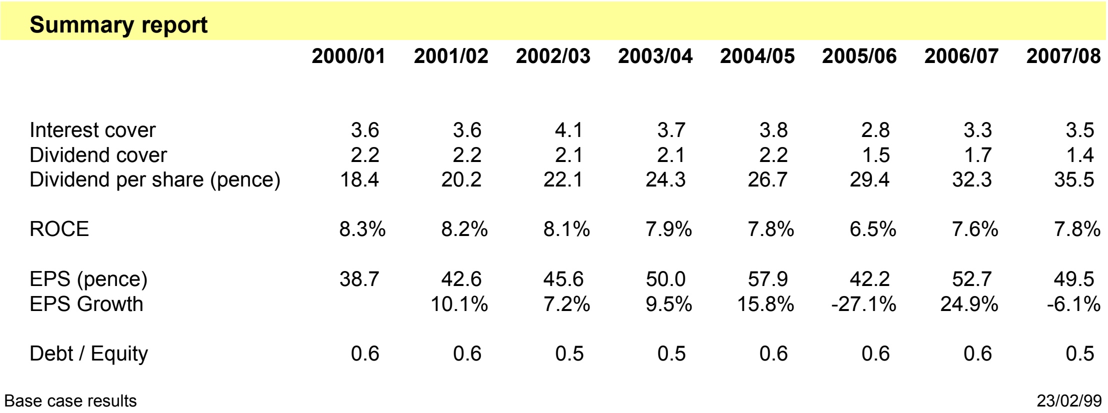


Figure 53: Example of reconciliation table graph

A diagram of this sort can be used to analyse the difference between any two model scenarios. It is particularly useful when there are a number of differences in the assumptions for the two versions of the model and you want to understand the importance of each changed assumption.

Comparator models

A comparator model is the quickest way to understand the effect of a single change on a wide range of model results. Consider the example report below.



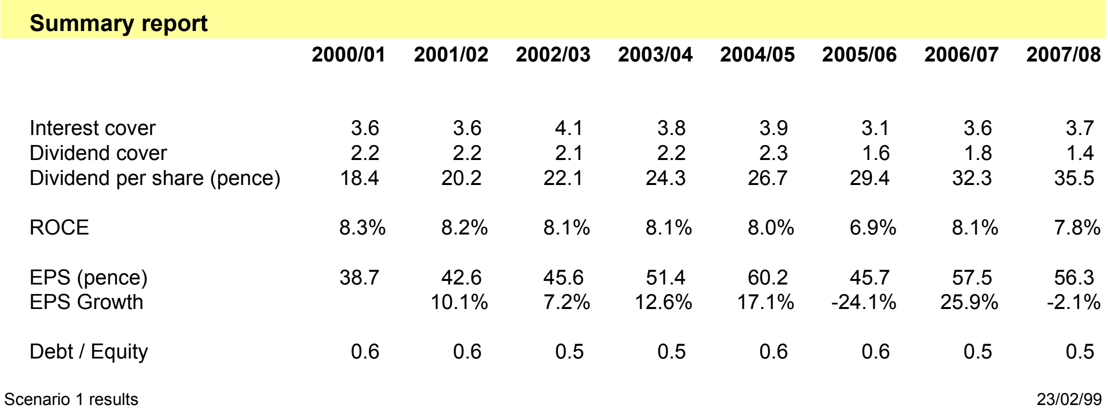
Summary report

	2000/01	2001/02	2002/03	2003/04	2004/05	2005/06	2006/07	2007/08
Interest cover	3.6	3.6	4.1	3.7	3.8	2.8	3.3	3.5
Dividend cover	2.2	2.2	2.1	2.1	2.2	1.5	1.7	1.4
Dividend per share (pence)	18.4	20.2	22.1	24.3	26.7	29.4	32.3	35.5
ROCE	8.3%	8.2%	8.1%	7.9%	7.8%	6.5%	7.6%	7.8%
EPS (pence)	38.7	42.6	45.6	50.0	57.9	42.2	52.7	49.5
EPS Growth		10.1%	7.2%	9.5%	15.8%	-27.1%	24.9%	-6.1%
Debt / Equity	0.6	0.6	0.5	0.5	0.6	0.6	0.6	0.5

Base case results 23/02/99

Figure 54: Example report – Base case

This model can then be used to consider many different alternative scenarios, such as the one below.



Summary report

	2000/01	2001/02	2002/03	2003/04	2004/05	2005/06	2006/07	2007/08
Interest cover	3.6	3.6	4.1	3.8	3.9	3.1	3.6	3.7
Dividend cover	2.2	2.2	2.1	2.2	2.3	1.6	1.8	1.4
Dividend per share (pence)	18.4	20.2	22.1	24.3	26.7	29.4	32.3	35.5
ROCE	8.3%	8.2%	8.1%	8.1%	8.0%	6.9%	8.1%	7.8%
EPS (pence)	38.7	42.6	45.6	51.4	60.2	45.7	57.5	56.3
EPS Growth		10.1%	7.2%	12.6%	17.1%	-24.1%	25.9%	-2.1%
Debt / Equity	0.6	0.6	0.5	0.5	0.6	0.6	0.5	0.5

Scenario 1 results 23/02/99

Figure 55: Example report – Alternative scenario

To understand the effect of the scenario on each of the measures, you need to compare the results from each of the two reports. This can be a slow process and it is easy to miss a significant change in a large table of numbers.

A comparator model is exceptionally simple. It consists of three worksheets. Each sheet contains an outline of the summary report. On the first sheet you place the base case results and on the second you place the scenario results. The third sheet calculates the difference between the first two sheets.

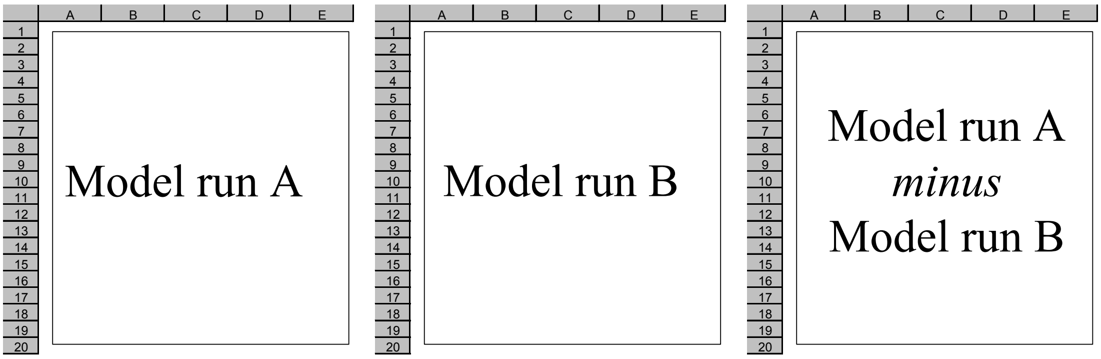
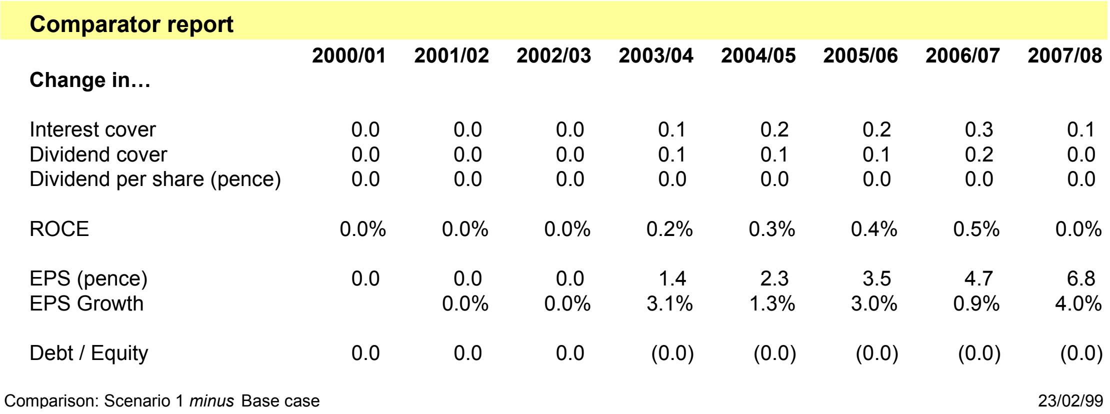


Figure 56: Layout of comparator model



Comparator report

Change in...

	2000/01	2001/02	2002/03	2003/04	2004/05	2005/06	2006/07	2007/08
Interest cover	0.0	0.0	0.0	0.1	0.2	0.2	0.3	0.1
Dividend cover	0.0	0.0	0.0	0.1	0.1	0.1	0.2	0.0
Dividend per share (pence)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ROCE	0.0%	0.0%	0.0%	0.2%	0.3%	0.4%	0.5%	0.0%
EPS (pence)	0.0	0.0	0.0	1.4	2.3	3.5	4.7	6.8
EPS Growth		0.0%	0.0%	3.1%	1.3%	3.0%	0.9%	4.0%
Debt / Equity	0.0	0.0	0.0	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)

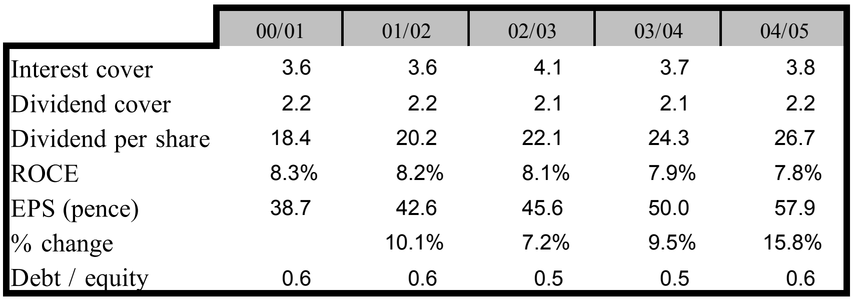
Comparison: Scenario 1 *minus* Base case 23/02/99

Figure 57: Example report – Comparator report

The comparator report shows you, at a glance, which lines in the report has changed and which have changed substantially. In this example, earnings per share is increased significantly, with a knock on effect on some of the other measures.

Run trees

Run trees are a technique for tracking the results from a number of interrelated model scenarios. To use a run tree you will need a simple model summary report, such as the example below. The run tree numbers can either be typed into the report or linked through to the underlying models.



	00/01	01/02	02/03	03/04	04/05
Interest cover	3.6	3.6	4.1	3.7	3.8
Dividend cover	2.2	2.2	2.1	2.1	2.2
Dividend per share	18.4	20.2	22.1	24.3	26.7
ROCE	8.3%	8.2%	8.1%	7.9%	7.8%
EPS (pence)	38.7	42.6	45.6	50.0	57.9
% change		10.1%	7.2%	9.5%	15.8%
Debt / equity	0.6	0.6	0.5	0.5	0.6

Figure 58: Example model summary report

A run tree, such as Figure 59 on the next page, shows the effect of a series of possible model scenarios on the summary report.

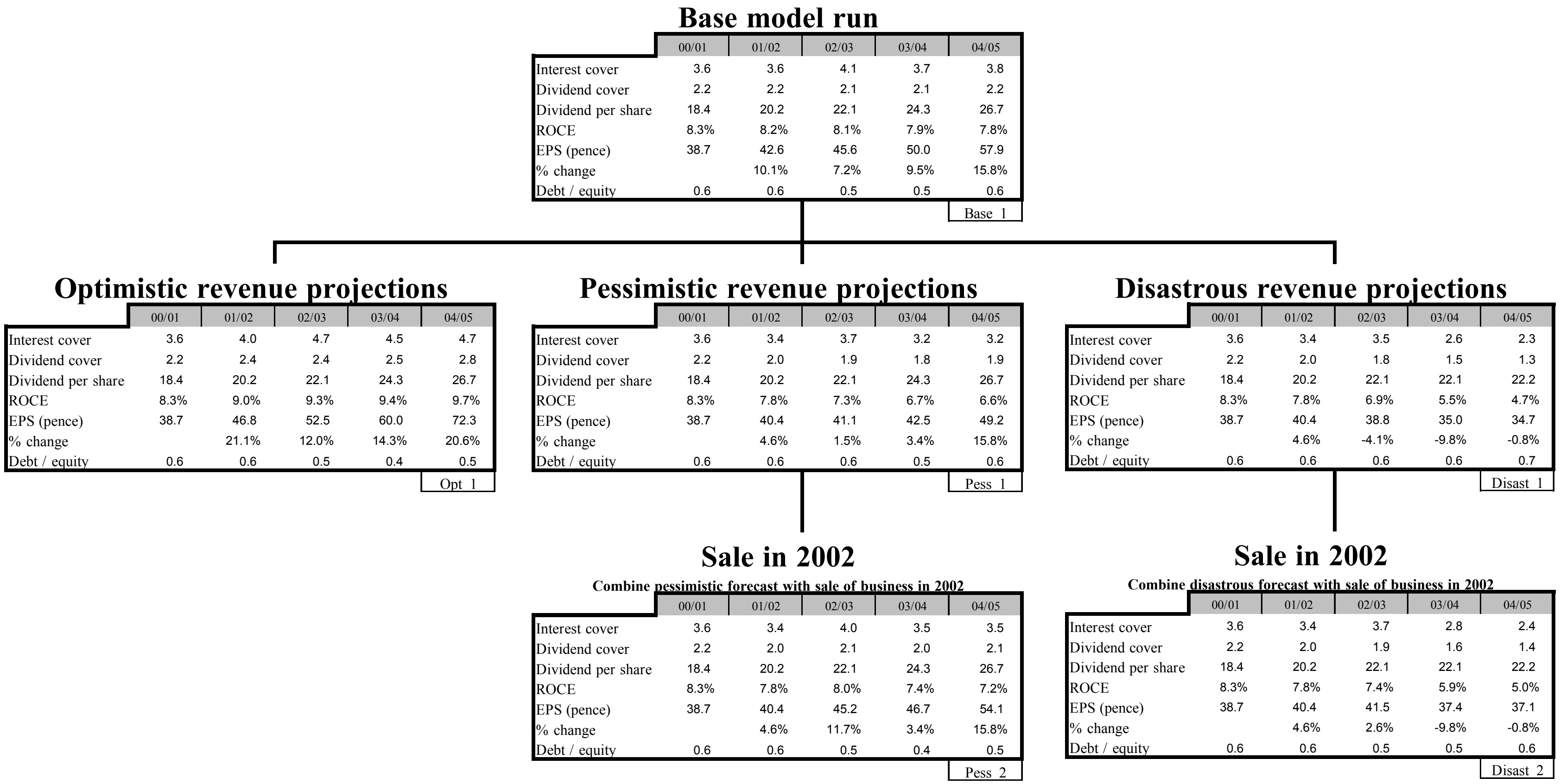
Each table on a branch of the tree represents a change from the base case. You can monitor the effect of a series of changes by following the model versions through the tree. When there are a large number of different model runs, a printer that can use large paper sizes is useful.

Monte Carlo analysis

One of the problems with carrying out sensitivity analysis is combining the individual sensitivities in the model, of which there may be many, in a way that realistically reflects the range of possible outcomes that may occur in practice. A simple approach that is often used is to set all the sensitivities to the worst case, and look at the results that this gives. However, by doing this the downside of the project is vastly over-stated; in reality, it is unlikely that everything will go wrong together.

Even if realistic combinations of sensitivities are used, simply applying these in the model does not provide any information about how likely, or probable, the resultant outcome is.

The technique of Monte Carlo analysis overcomes these problems: it allows the user to combine many sensitivities in many different ways, effectively running hundreds, or even thousands, of scenarios. The output shows the range of possible results, and how probable each is. This information enables the user to understand the effect that variations in the inputs have on the output: in other words, this method makes clear the risks involved in the project.



Base model run

	00/01	01/02	02/03	03/04	04/05
Interest cover	3.6	3.6	4.1	3.7	3.8
Dividend cover	2.2	2.2	2.1	2.1	2.2
Dividend per share	18.4	20.2	22.1	24.3	26.7
ROCE	8.3%	8.2%	8.1%	7.9%	7.8%
EPS (pence)	38.7	42.6	45.6	50.0	57.9
% change		10.1%	7.2%	9.5%	15.8%
Debt / equity	0.6	0.6	0.5	0.5	0.6

Base 1

Optimistic revenue projections

	00/01	01/02	02/03	03/04	04/05
Interest cover	3.6	4.0	4.7	4.5	4.7
Dividend cover	2.2	2.4	2.4	2.5	2.8
Dividend per share	18.4	20.2	22.1	24.3	26.7
ROCE	8.3%	9.0%	9.3%	9.4%	9.7%
EPS (pence)	38.7	46.8	52.5	60.0	72.3
% change		21.1%	12.0%	14.3%	20.6%
Debt / equity	0.6	0.6	0.5	0.4	0.5

Opt 1

Pessimistic revenue projections

	00/01	01/02	02/03	03/04	04/05
Interest cover	3.6	3.4	3.7	3.2	3.2
Dividend cover	2.2	2.0	1.9	1.8	1.9
Dividend per share	18.4	20.2	22.1	24.3	26.7
ROCE	8.3%	7.8%	7.3%	6.7%	6.6%
EPS (pence)	38.7	40.4	41.1	42.5	49.2
% change		4.6%	1.5%	3.4%	15.8%
Debt / equity	0.6	0.6	0.6	0.5	0.6

Pess 1

Disastrous revenue projections

	00/01	01/02	02/03	03/04	04/05
Interest cover	3.6	3.4	3.5	2.6	2.3
Dividend cover	2.2	2.0	1.8	1.5	1.3
Dividend per share	18.4	20.2	22.1	22.1	22.2
ROCE	8.3%	7.8%	6.9%	5.5%	4.7%
EPS (pence)	38.7	40.4	38.8	35.0	34.7
% change		4.6%	-4.1%	-9.8%	-0.8%
Debt / equity	0.6	0.6	0.6	0.6	0.7

Disast 1

Sale in 2002

Combine pessimistic forecast with sale of business in 2002

	00/01	01/02	02/03	03/04	04/05
Interest cover	3.6	3.4	4.0	3.5	3.5
Dividend cover	2.2	2.0	2.1	2.0	2.1
Dividend per share	18.4	20.2	22.1	24.3	26.7
ROCE	8.3%	7.8%	8.0%	7.4%	7.2%
EPS (pence)	38.7	40.4	45.2	46.7	54.1
% change		4.6%	11.7%	3.4%	15.8%
Debt / equity	0.6	0.6	0.5	0.4	0.5

Pess 2

Sale in 2002

Combine disastrous forecast with sale of business in 2002

	00/01	01/02	02/03	03/04	04/05
Interest cover	3.6	3.4	3.7	2.8	2.4
Dividend cover	2.2	2.0	1.9	1.6	1.4
Dividend per share	18.4	20.2	22.1	22.1	22.2
ROCE	8.3%	7.8%	7.4%	5.9%	5.0%
EPS (pence)	38.7	40.4	41.5	37.4	37.1
% change		4.6%	2.6%	-9.8%	-0.8%
Debt / equity	0.6	0.6	0.5	0.5	0.6

Disast 2

Figure 59: Example of a run tree

To carry out a Monte Carlo analysis, you need to:

- understand what the key uncertainties and risks are;
- specify probability distributions on the inputs that are uncertain. By assuming probability distributions, such as the examples in Figure 60, you are defining not just the range of likely outcomes but the likelihood of each arising; and
- estimate the correlation between the input assumptions, for example between your estimates for interest rates and inflation. Failure to include correlation where appropriate will lead to substantially underestimating the variability of the results.

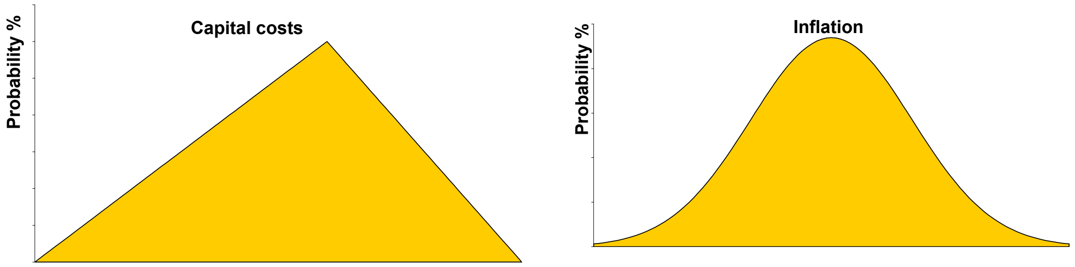


Figure 60: Example input distributions

The Monte Carlo simulation produces a probability distribution of the output from the model that shows the likelihood of occurrence of all values of the output, such as Figure 61.

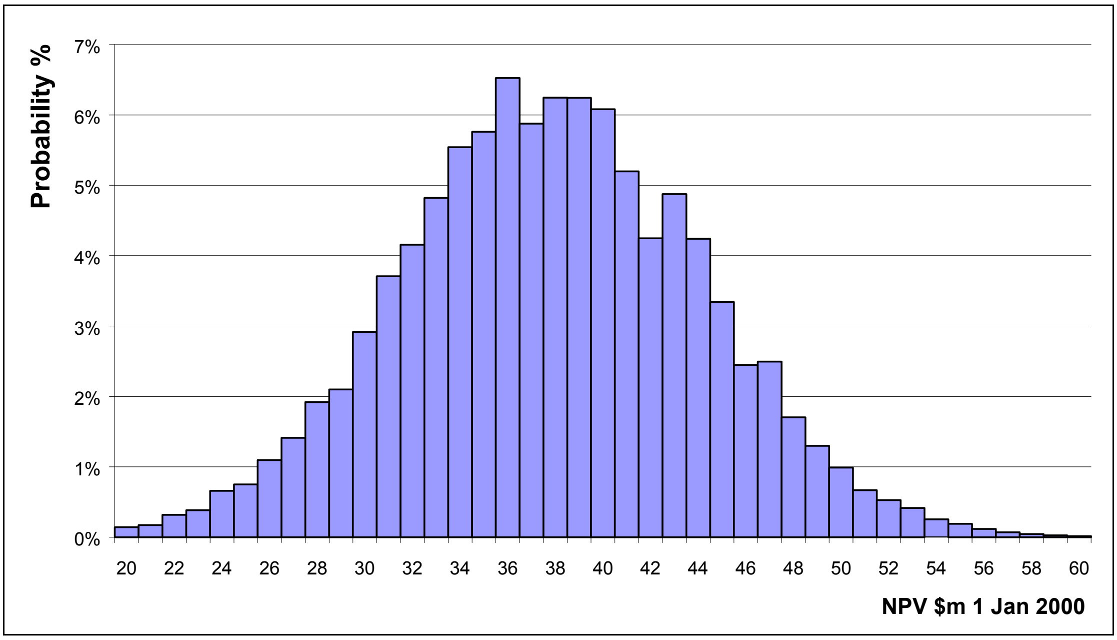


Figure 61: Example output distribution

You can use these output graphs, and the statistical measures produced by the add-in packages, to establish what the most likely result of the project is, what the range of possible outcomes could be, and the chance of failing to meet investment criteria, for example failing to achieve a target NPV. These results can help to understand the risks that the project involves, and to investigate the effects of risk mitigation strategies.

Appendix A includes details of Excel add-in packages for Monte Carlo analysis.

Summary of techniques for running scenarios

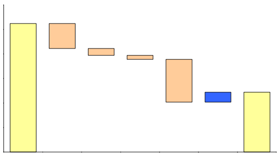
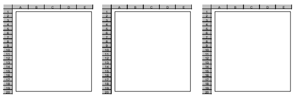
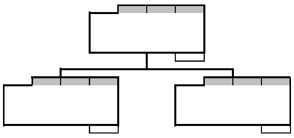
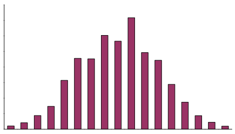
<table>
<tr><th>Technique</th><th>Use to…</th></tr>
<tr><td>Data tables
<table><tr><th>Scale advertising income by</th><th>NPV at 1/1/1999</th><th>Rate of return</th><th>Max. funding requirement</th><th>Cash flow positive in year</th></tr><tr><td>-10%</td><td>135.2</td><td>23.4%</td><td>125.6</td><td>2002</td></tr><tr><td>-5%</td><td>164.0</td><td>25.4%</td><td>124.6</td><td>2002</td></tr><tr><td>Base case = 0%</td><td>192.9</td><td>27.4%</td><td>123.6</td><td>2002</td></tr><tr><td>5%</td><td>221.5</td><td>29.3%</td><td>122.6</td><td>2002</td></tr><tr><td>20%</td><td>278.6</td><td>32.9%</td><td>120.5</td><td>2001</td></tr></table></td><td>• understand the sensitivity of the results to one or two of the key input assumptions</td></tr>
<tr><td>Reconciliation tables and graphs
</td><td>• track the effect of a series of incremental changes to the model results
• analyse the differences between two model scenarios, especially when the two scenarios have a number of distinct differences
• separate the effect of changes to model inputs from corrections to the model code</td></tr>
<tr><td>Comparator models
</td><td>• focus quickly on the changes to a report caused by a scenario
• highlight those results that have changed significantly, particularly in lengthy report such as a set of financial statements</td></tr>
<tr><td>Run trees
</td><td>• present scenarios which have both alternative options and incremental changes
• combine with comparator models to see the effect of each increment</td></tr>
<tr><td>Monte Carlo analysis
</td><td>• analyse the effect of distributions of input assumptions, rather than just fixed values or ranges
• understand the combined effect of a number of different risks</td></tr>
</table>

Figure 62: Summary of techniques for running scenarios

Graphs

Graphs allow faster assimilation of information than a page of numbers. Graphs are frequently valuable as a way to introduce information, but usually need to be supported by the actual numbers. Bear in mind that different people in your audience may have different needs, those who just want to understand the basic trends may want to see the graphs while those who need to understand more detail are likely to want to see the numbers behind it.

Consider the following example of widget sales figures projected for the next few years.

	1999	2000	2001	2002	2003
Blue	60	70	80	80	80
Green	15	20	20	20	20
Red	10	10	10	5	5
Advanced	5	10	15	20	25
Bespoke	0	5	15	25	40

Figure 63: Widget sales in 000s, 1999 to 2003

To present the information in this table more quickly, we produced the graph below.

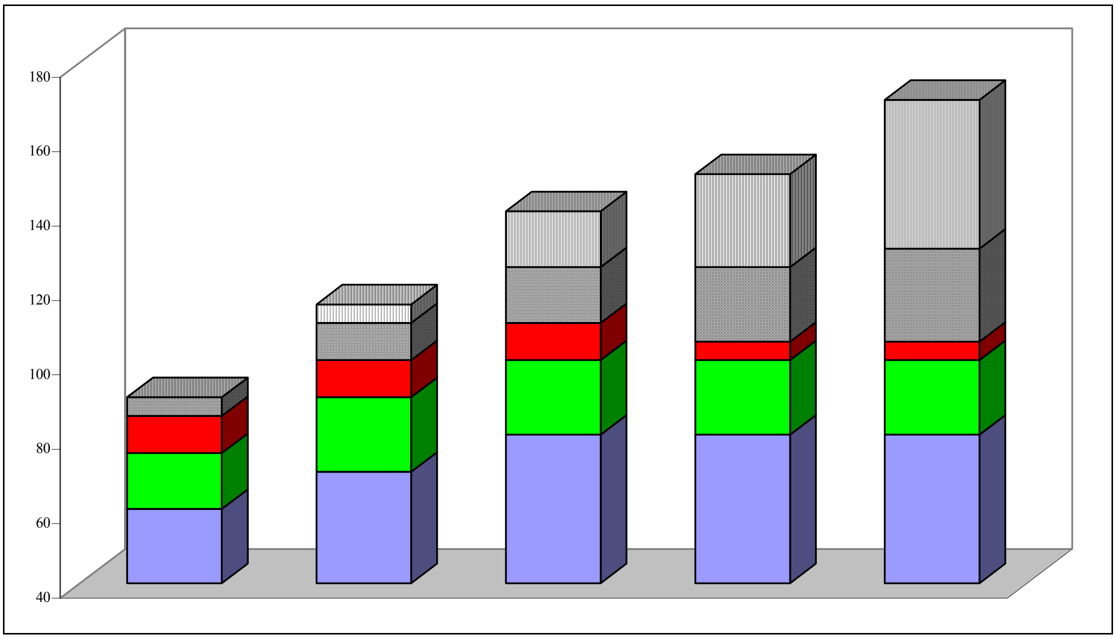


Figure 64: Example of a bad graph

This is a dreadful use of a graph, and it is unlikely that anyone would ever try to present such a picture. It does, however, highlight a number of the design points to bear in mind when using graphs.

Labelling and descriptions

- Label clearly with title, axes description and legends. Our example gives no indication of which block represents sales of which type of widget.
- Communicate a message, not just the numbers - use labels to tell the reader what is actually happening in the results.

Use of colour

- Distinguish between different blocks with contrasting colours that will still appear different if the graph is reproduced in black and white.
- Blue, green and red widgets have been shown here in blue, red and green. Unfortunately it does not work in black and white.

Axes

- Beware of graphs that truncate the y-axis by showing the bottom of the scale as anything other than zero. They distort the results by exaggerating the effect of small changes.
- Our example starts the y-axis at 40, under-representing the proportion of sales made up by blue widgets. This approach should not be used for bar charts, especially stacked bar charts of this sort.

Use of stacked bar charts

- In a stacked bar chart, do not use more than three components.
- It is very difficult to track the changes in any of the middle components in our graph.

3D graphs

- Three dimensional graphs may be striking, but they are usually much more difficult to interpret than their two dimensional equivalent.
- The 3D blocks in our example make it harder to understand how sales have changed.

Fonts

- Make the text legible, using different styles of text sparingly for effect.
- In our example, the axis labels are rather small. Sans serif fonts are sometimes clearer for graph labels.

Bearing in mind some of the principles, consider the alternative graph below.

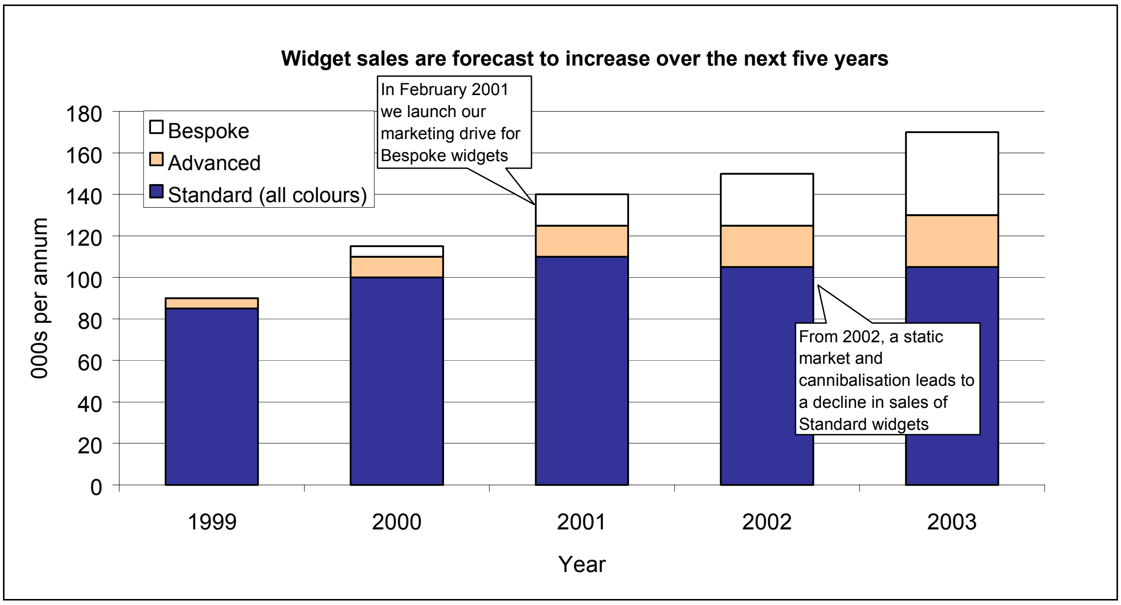


Figure 65: Example of a better graph

The real test for the effectiveness of a graph is what useful information it actually conveys. What can we actually say about widget sales using this graph more easily than we could before?

This graph tells us that while standard widgets make up the vast majority of sales, any growth in the next few years will be limited, while other widgets (especially bespoke widgets) will become increasingly significant. If it is important to understand how standard widget sales break down into the different colours, you could produce another graph showing this.

Another common mistake is to use an inappropriate type of graph for the message you want to present. The table below summarises the strengths of weaknesses of the major graph types.

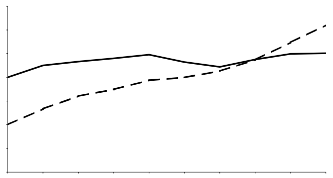
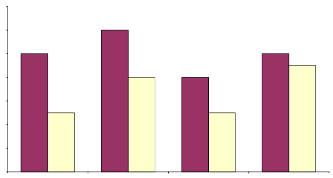
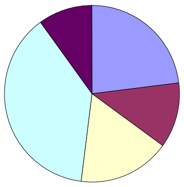
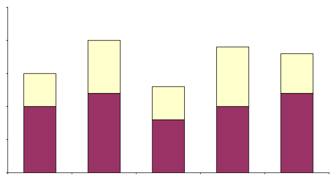
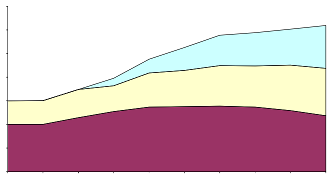
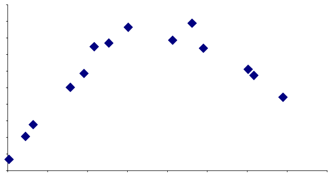
Type of graph	**Use to…**	**For example …**
Line graph 	present continuous data, especially movements over a large number of time periods compare the behaviour of a large number of variables, especially when they are close together and would be difficult to distinguish between in a bar chart	unemployment figures 1970 to 1998 GDP per capita for selected countries 1970 to 1998
Bar chart 	present discrete data compare the behaviour of a small number of different variables present movements over a short period of time, when a line graph may appear rather awkward	product sales by region sales of three types of product by region product sales 1999 to 2003
Pie chart 	show the break down of a single variable into its component parts, particularly to emphasise that together the parts add to 100%, … but never use two pie-charts side by side for comparison	mode of journey to school 11-16 year olds
Stacked bar chart 	show how the breakdown of a variable into its component parts fluctuates, provided that the number of parts is limited indicate the ranking in an example, by sorting the bars	mode of journey to school 11-16 year olds by year 1994 to 1999 productivity by country, sorted
Area graph 	show the breakdown of a continuous variable into its constituent parts	uptake of multi-channel television by distribution method, 1990 to 2020
X-Y (or scatter) graph 	understand the relationship between two variables	crop yield versus annual rainfall

Figure 66: When to use different graph types

"Putting it all together"

This chapter has focussed on some of the different ways that you can present information produced by your model. Now we will consider how to bring these ideas together, producing a clear and concise message.

Run packs

A run pack is a presentation of a collection of model results. It could include:

- an executive summary of the key messages and results contained in the pack;
- reports, tables and graphs from the model summarising the key results and supported by text descriptions of the results;
- details of the key model input assumptions;
- an exploration of model sensitivities and scenarios using any appropriate techniques including those we introduced in this chapter;
- supporting print outs of the model results and calculations; and
- next steps, including recommended decisions or options for further model development and analysis.

When you are putting together a run pack, make sure that you understand the readership for the document. Frequently you may have a range of readers, from those who will only read a suitable summary to those who want to pore over the model calculations.

Presentations

Run packs can be written as documents to be read individually, but it is often more powerful to use a combination of a written document and a presentation to get your message across.

By presenting results to an audience you can build up to a set of conclusions from the model analysis, using written material as a support for those who want more detail.

Since the advent of affordable on-line projection equipment, it has become feasible to use a model in front of an audience. This technique can be very powerful, but it will put the design and ease of use of your model under the microscope. Again, it is important to understand your audience and how much of the detail of the model they want to see.

A model which is easy to understand and looks impressive on screen, together with a user who can quickly run what-if questions is perhaps the best demonstration of the benefits of a best practice model.

Staying in control of your model

Used badly, models often deteriorate. They can start well, producing useful results, but as changes are made over time, errors are introduced, the functionality changes and the results in the model change - and frequently nobody is quite sure why.

Staying in control of your model is one of the greatest challenges for the model user.

When the model is being used for a specific deal or transaction, additional changes often pile up just before the deadline date. By staying in control of your model, you can make sure that the model results can continue to be relied upon right up to the deadline.

When the model is developed for long term use, staying in control can maintain the credibility and usefulness of your model and can extend its overall lifespan.

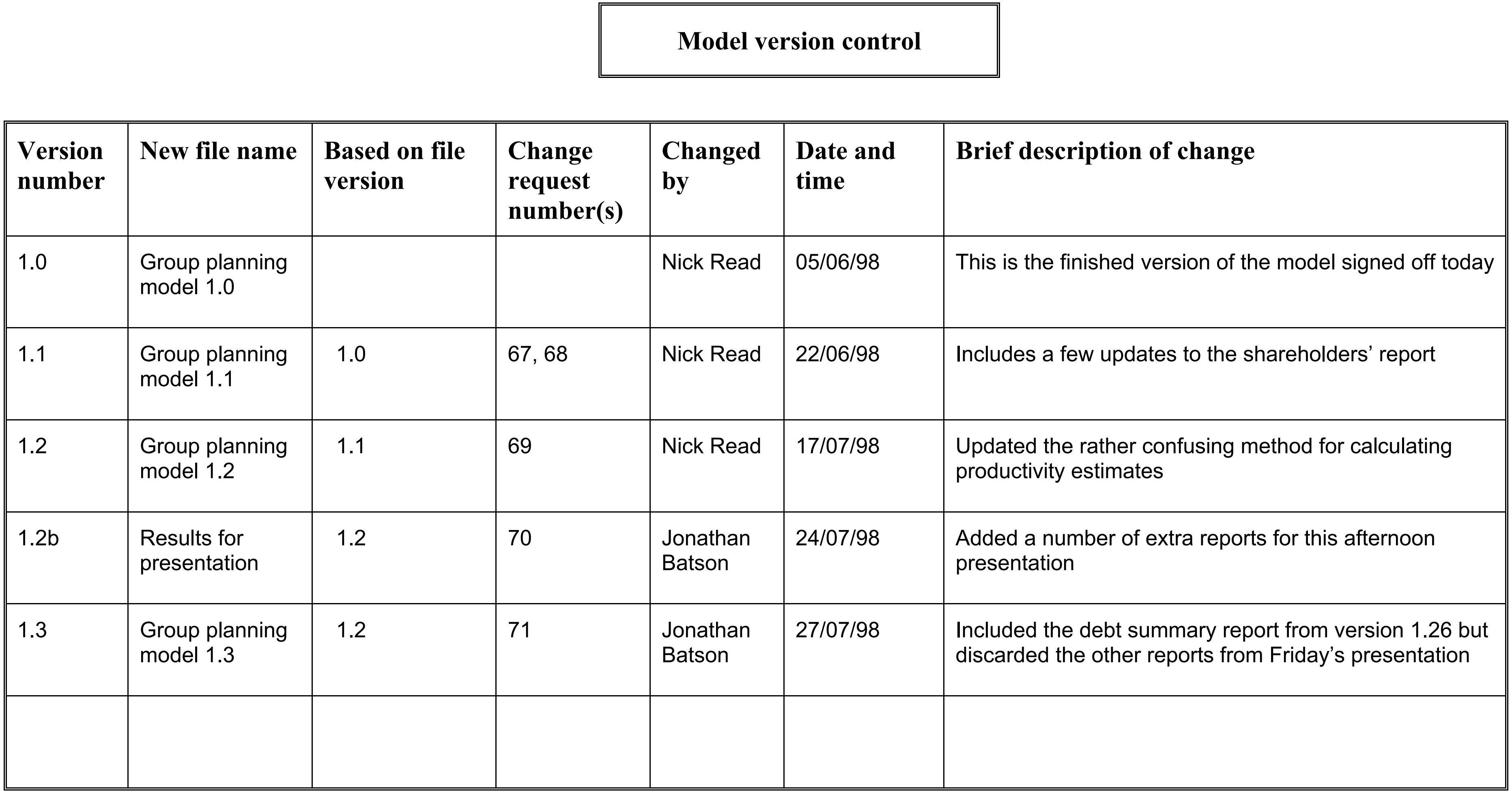
Model version control

As the business evolves (or your understanding of it develops) the model will usually need to change. If you fail to keep track of changing model versions, it is easy to mix up old and new models. As a result, you can present results that do not include the latest corrections to the model or waste time typing the latest input assumptions into an out of date version of the model.

To keep control of the model version, we recommend keeping two simple records: model change requests and model change control.

Change requests were introduced in the test chapter, and an example change request form is on page 7-71. Using change request forms to record all changes to the model provides a record of all model changes, why the change was made and who did the work. It also provides a record of when the model was last tested.

To keep track of all of the change request forms, we recommend using a model change control form, such as the example below. The change control form keeps an at-a-glance record of all versions of the model. Most importantly, it will always identify the latest version of the model so that you will never mix up old and new versions.



Model version control

Version number	New file name	Based on file version	Change request number(s)	Changed by	Date and time	Brief description of change
1.0	Group planning model 1.0			Nick Read	05/06/98	This is the finished version of the model signed off today
1.1	Group planning model 1.1	1.0	67, 68	Nick Read	22/06/98	Includes a few updates to the shareholders' report
1.2	Group planning model 1.2	1.1	69	Nick Read	17/07/98	Updated the rather confusing method for calculating productivity estimates
1.2b	Results for presentation	1.2	70	Jonathan Batson	24/07/98	Added a number of extra reports for this afternoon presentation
1.3	Group planning model 1.3	1.2	71	Jonathan Batson	27/07/98	Included the debt summary report from version 1.26 but discarded the other reports from Friday's presentation

Figure 67: Example model change control form

Run control

When you are running numerous scenarios through a model, it can be difficult to keep track of all of the different versions of results. If you fail to keep control of the different runs of the model, it is difficult to remember which set of input assumptions led to which conclusion, and when someone asks about a set of model results you find that you are unable to recreate them.

The best way to keep control of model runs depends on the size and complexity of the model, bigger models may require more detailed control. For most models, a run control form, such as Figure 68 on next page, should suffice.

A run control form of this type keeps a record of all of the model runs that you do, so that if you are asked, you have a good idea of what you did when you created the run.

This type of run control form does not, except for the simplest of runs, provide sufficient information for you to recreate the run, or list all of the input assumptions it uses. To provide this more detailed record of model runs, you might file a more detailed list of all of the assumptions used for each run, perhaps by printing the input sheet of the model. (Provided that you designed your model to keep all of the input assumptions together.)

How to store model runs

Our example run control form includes a column for the File name of the run. The most common way to store the files used to create each model run is to make a copy of the model used to produce it. This means that:

- if you make changes to the model, previous runs will be based on an out of date version and you may have to recreate them; and

- if you do numerous runs with a large model, a large amount of storage space is required.

For models that will be used many times, a useful alternative method is to store just the model inputs, without storing the model itself. This can be done by copying the input area from the model into a separate file, which you can retrieve at a later date and copy back into the model. To speed up this process and reduce the risk of human error, this task is particularly well suited to automation by a simple macro.

If you plan to use this method for storing runs, remember that:

- it becomes absolutely essential that you follow the first golden rule of spreadsheet design: separate your inputs and calculations. All of your inputs must be contained in the saved input block if you are to use this to recreate the model results; and

- any change to the master copy of the model that involves inserting new rows and columns onto the input sheet will invalidate all of your saved model files.

Model run control

Scenario name	Based on model version	File name	Runs by	Date and time	Description of model run
Base case forecast	1.0	Group planning model 1.0 - Base case	Nick Read	05/06/98	Base case estimate based on next year's budget
Worst case analysis	1.0	Group planning model 1.0 - Worst case	Nick Read	08/06/98	Worst case estimate from the draft budget plan
Acquisition plan	1.0	Group planning model 1.0 – Acquisition plan	Jonathan Batson	12/06/98	Estimates the effect of the acquisition plan we are discussing tomorrow
Updated base case forecast	1.1	Group planning model 1.1 - Base case	Nick Read	23/06/98	Base case estimate using the updated version of the model

Figure 68: Example run control form

Why use control forms?

At first sight, the control forms described in this document may seem rather cumbersome to use. However, used appropriately, the time required each time you change the model is fairly short and will repay itself in saved time resolving queries about previous results produced by the model. Being able to answer questions reliably and consistently also helps to build the credibility of the model and its results.

Documentation

If a model is to be used by someone other than the model developer, some form of documentation will probably be required to hand the model over. Good documentation:

- allows another person to use a model and understand what it is doing even if the model has been out of use for some time;
- reduces the risk that the model falls into disuse because only one or two people know how to use it or trust its results; and
- reduces the number of irritating interruptions for the model developer, after handing over responsibility for the model.

There are three types of documentation that are commonly produced: a specification, user guide and technical documentation.

Specification

The specification document was introduced earlier. It is still a useful document at this stage because, written well, it is now an accurate definition of how the model works. For anyone who might need to understand the model in future, an accurate specification is much easier to understand than relying on a complicated model or the memory of the model developer. Without an up to date specification it is much more difficult to trust that the model is still doing what is required of it.

User guide

When someone other than the model developer will need to use the model, a user guide explains how to do it. A user guide will typically include:

- a description of the purpose of the model;
- operating instructions, including a description of the regular steps required to produce model results; and
- a case study which gives an example of how the model is used.

When the model will be used by a number of different people, you can support the user guide by including training sessions to explain the operation of the model.

Technical documentation

Technical documentation is useful for a model that might need to be updated or expanded in future. It is particularly useful when the model developer will not be available for future changes, but even when this is not the case the documentation can serve as a aide-mémoire for the more complicated parts of the model.

Technical documentation may include:

- technical notes on the design of the model;
- a change log, describing all changes to the model since it was originally written; and
- an explanation of the work that would be required to expand or update the model for likely future changes, for example if the business being modelled were to grow.

Chapter summary **Use**

- A model user is not a model operator: don't just produce model print outs, produce valuable information that will help people understand the issues and make the right decisions.
- A best practice model is a valuable tool, make the most of it to gain competitive advantage.
- Techniques for presenting results can hugely increase the usefulness of a model, especially when presenting the effects of a number of different scenarios.
- Control forms can help you keep track of different versions of the model and understand how changes in your assumptions have changed the results from the model.

9 Appendix A: software packages

Spreadsheet auditing

Model testing is much more effective if you use spreadsheet auditing software to analyse the model layout.

Spreadsheet Professional is a commercial package which contains of number of additional features to assist model development. A number of these features tie in to the specific recommendations in the guide. For example, Spreadsheet Professional will produce:

- formula maps of the type described in the test chapter;
- reasonable calculation tables from a spreadsheet model, provided that some reasonable design principles are followed; and
- data input tables similar to those introduced in the specify chapter.

Spreadsheet Professional is available from Spreadsheet Innovations Ltd in the UK on +44(0)20 7424 0101. There are separate versions for Excel 97, Excel 95 and Lotus 1-2-3 (version 4 onwards).

Other spreadsheet auditing software is available, for example Southern Cross Software's Spreadsheet Detective package (www.uq.net.au/detective) contains much the same features as Spreadsheet Professional.

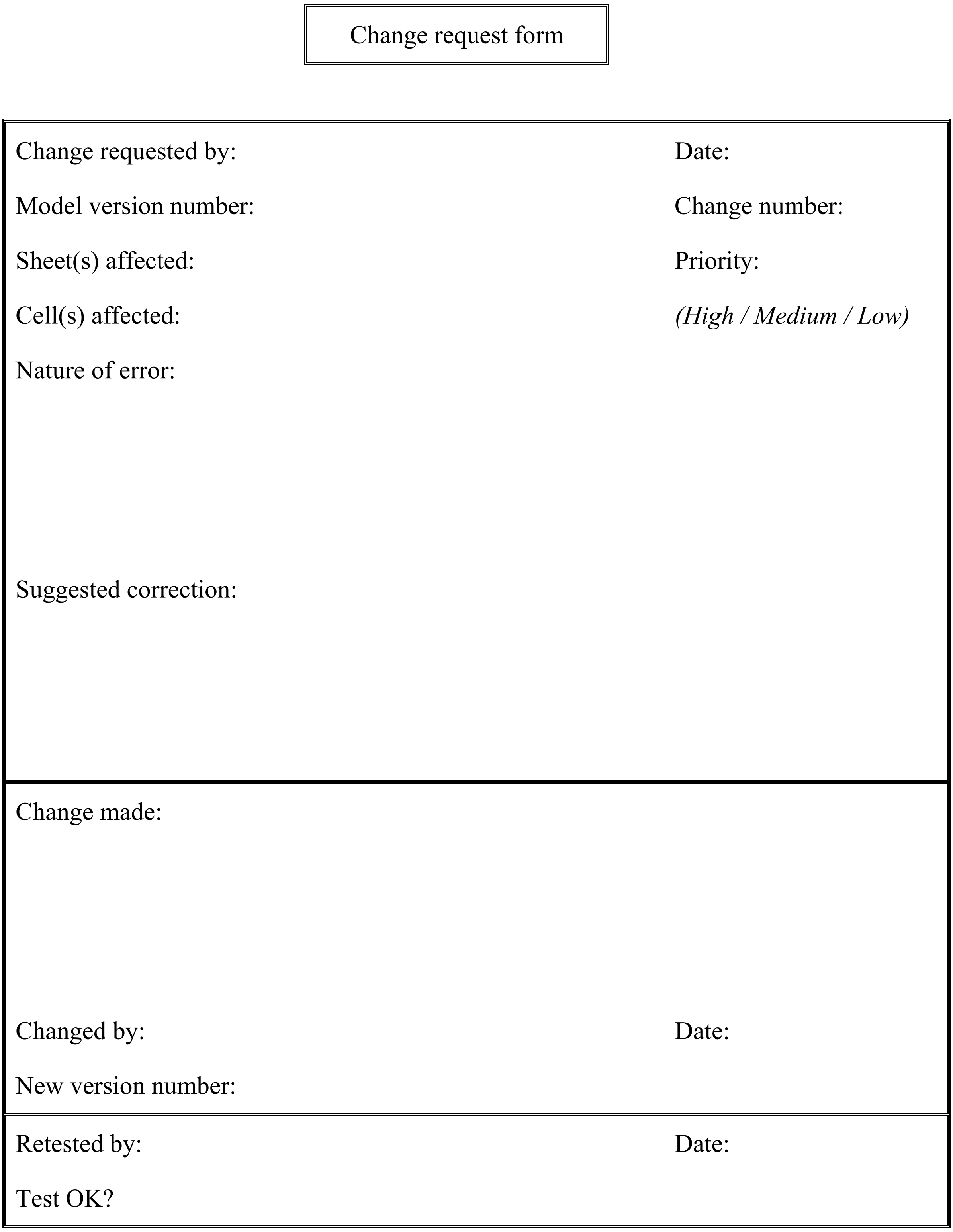
Monte Carlo analysis

In Excel 97, there are add-in software packages available which will automate the process of running a Monte Carlo simulation. These allow you to:

- provide inputs in the form of estimated distributions;
- specify correlation between input assumptions; and
- sample repeatedly from input distributions to produce estimated output distributions.

Packages available include Crystal Ball (www.decisioneering.com) and @Risk (www.palisade.com).

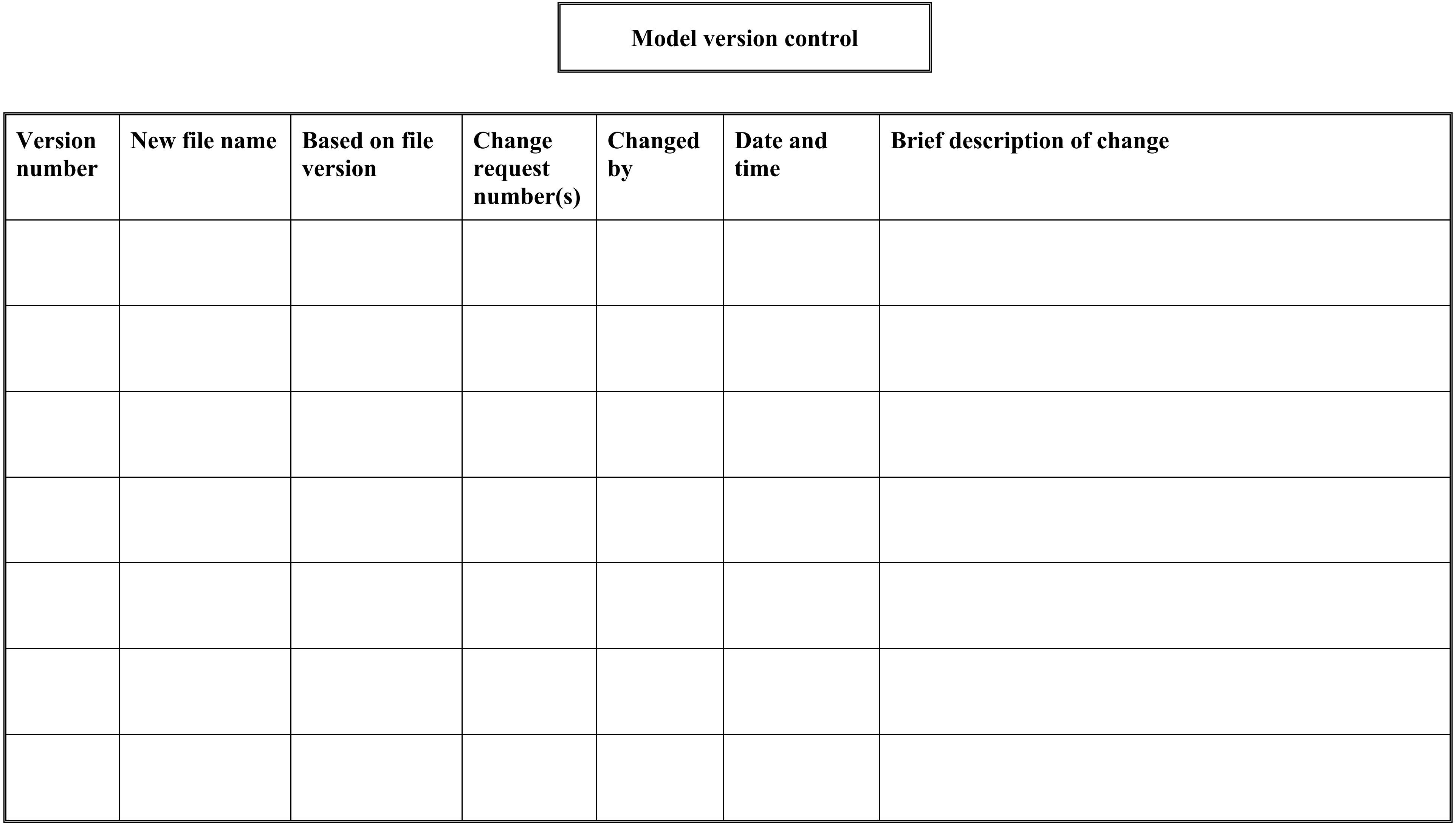
10 Appendix B: sample control forms



Change request form

Change requested by:	Date:
Model version number:	Change number:
Sheet(s) affected:	Priority:
Cell(s) affected:	*(High / Medium / Low)*
Nature of error:	
Suggested correction:	
Change made:	
Changed by:	Date:
New version number:	
Retested by:	Date:
Test OK?	

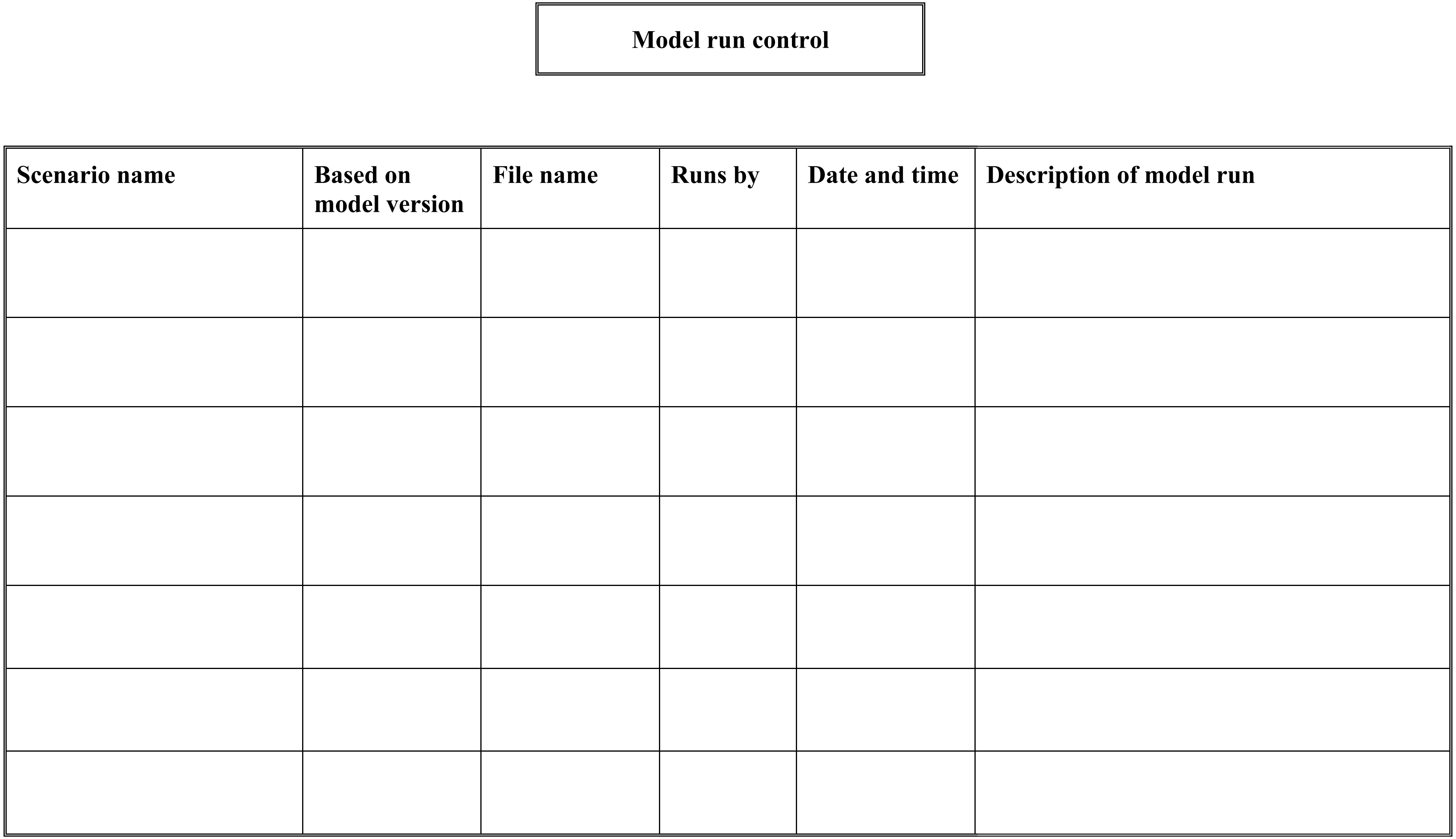
Figure 69: Sample change request form



Model version control

Version number	New file name	Based on file version	Change request number(s)	Changed by	Date and time	Brief description of change

Figure 70: Sample model change control form



Model run control

Scenario name	Based on model version	File name	Runs by	Date and time	Description of model run

Figure 71: Example run control form

For further information

If you would like to know more about the services Business Dynamics provide and how we can help your business, please contact:

Gavin Potter, IBM Business Consulting Services, 1 London Bridge, London, SE1 9BD

Telephone: +44 (0)20 7968 5401

Fax: +44 (0)20 7968 8667

e-mail: gavin.s.potter@uk.ibm.com