Choosing a Lifecycle Model

1 Planning and managing your project

You should read this guide before starting TMA 01.

Choosing an approach to planning and managing your project is one of the main tasks you need to undertake for TMA 01. The aim of this guide is to help you make and justify that choice. Subsequent assessments will require you to revisit the plan, and your final report (the EMA) should contain an evaluation of your chosen lifecycle model in the light of your actual experience.

In your studies so far you are likely to have met at least one account of how projects can be planned and managed. This guide doesn't reproduce all of these different accounts in detail, instead it gives an outline of a number of different approaches and then focuses your attention on choosing an approach which best suits your chosen project. In this way, it supports you directly in heading towards achieving one of your most important learning outcomes: 'Plan and organise your project work appropriately, justifying your approach'.

At a very high level of abstraction, a remarkable number of computing projects can be considered as having the same four activities, each of which addresses a specific question:

- **Analysis** – What is the problem and what must a solution achieve?
- **Design** – How can the elements of a proposed solution, or of a proposed research process, be synthesised?
- **Implementation** – What needs to be done to deliver on the design and what is the best way of doing it?
- **Evaluation** – How can the performance of the solution, or the application of research ideas, be assessed in terms of solving the problem addressed?

(Different project management methods may use different terminology, further subdivide the above activities, or even add additional activities, such as 'system maintenance'.)

The four activities of analysis, design, implementation and evaluation can be considered as the 'building-block' practices that turn a problem into a solution.

So if projects are all the same at some level, what makes them different? Probably the most important factor is *uncertainty*. For some projects it is easy to state what the problem is, what the solution should look like, and how the solution can be arrived at and evaluated. For other projects the problem may be very poorly understood; there may be very different perspectives on what the solution should look like, and even when the nature of the solution is imagined it may be far from certain that known computational techniques can deliver the level of performance required; and finally, methods for evaluating the outcome may be difficult or even impossible to define, at least at an early stage. Of course, most projects are somewhere between these two ends of the uncertainty spectrum.

Project management is an *additional* activity that structures the four basic activities over the lifetime of the project, producing what is commonly called a *project lifecycle*. A lifecycle is a description of all the activities and outputs in the life of a project, and the sequence in which they happen. The choice and design of a project lifecycle needs to support effective and efficient progress towards the project outcome.

It is very easy to let the words ‘effective’ and ‘efficient’ pass without notice – after all, who would ever argue for ineffective or inefficient project management! But, given that they are often used almost as one word, it is worth paying some attention to what they mean and what the difference is between them. Much can be said on this, but for now we can recognise that an effective project is one that solves the problem it sought to address. Note that this is not the same as simply delivering the final output of the project, such as some new software application, a database, or a design for an interface. An efficient project is one that minimises the net cost involved, where ‘cost’ is a sophisticated measure that captures many different elements both negative (monetary cost of equipment, accommodation, energy, salaries, etc.) and positive (new skills, organisational structures and cultures acquired, opportunities for further exploitation, etc.).

Effectiveness and efficiency can be seen as going hand-in-hand, but the most effective solution may not be the most efficient, and vice versa. In reality, it is often necessary to trade one off against the other. *For projects that involve a lot of uncertainty, especially in relation to one or two crucial components of the overall solution, the need for efficiency points directly to the desirability of tackling those components as early as is feasible.*

2 Project lifecycle models

In this section we outline some of the main classes of lifecycle model that you should consider. In looking at the different lifecycle models, think about the way in which each model is more or less suited to the relative uncertainty associated with a project and its different phases. Think too about how each model might contribute towards overall efficiency.

2.1 ‘No going back’ – the classic waterfall lifecycle

[
The waterfall lifecycle model is commonly credited to Winston Royce (1970), although he did not use the term ‘waterfall’; an account can be found in the Wikipedia entry for ‘Waterfall model’.
]

Arguably something of a myth (nothing is ever this straightforward), the classic waterfall lifecycle suggests that each of the activities in a project (analysis, design, implementation and evaluation) are undertaken in strict sequence and that once a task is completed, there should be little if any need to go back. The rationale is that you can’t, or at least shouldn’t, design something until you know exactly what you want, that you can’t implement anything until you have the design, and that there is nothing to test until implementation is complete (Figure 2.1).

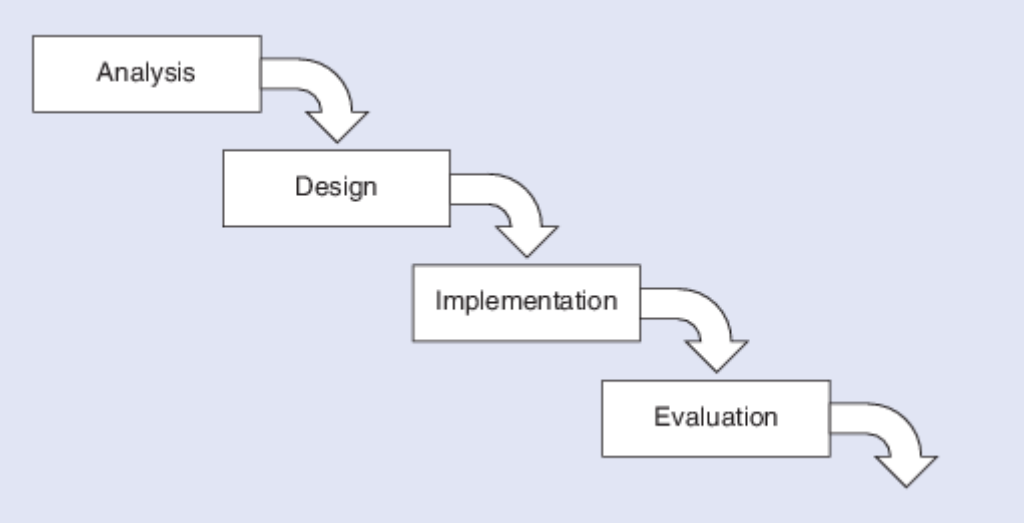

Figure 2.1 The classic ‘no going back’ waterfall lifecycle

Nonetheless, many projects can be planned broadly along these lines. This is especially so if the nature of the problem is well defined and there is a lot of prior knowledge and experience. In computing, many routine software development projects or relational database projects can be thought of as falling into this category.

The classic waterfall model can be used if you are satisfied that a number of conditions apply, some of the most important being:

1. There is a very *limited number of stakeholders* and they are all in explicit agreement about what the project is required to deliver and most, if not all, believe they stand to gain from the outcome.
2. The task is *small,* delivering limited additional functionality from limited code that can be tested comprehensively and preferably exhaustively.
3. It is *routine*, that is you are familiar with just about all aspects of the problem and there is a record of past success.
4. It will have a very *limited and clearly defined impact* on existing or otherwise related systems.

2.2 Iteration in project lifecycles

Somewhat nearer reality than the myth of classic waterfall development are lifecycle models featuring *iteration* – repeating one or more activities until adequate progress has been made. It would be a mistake to think of iteration as just ‘trial and error’: the point of iteration is to manage the process so that the ‘trials’ get better and the ‘errors’ get smaller. Some iterative lifecycles are relatively strict about the order in which tasks are revisited, others are much less strict.

Iterative waterfall model

In the iterative version of the waterfall model, each of the four steps – analysis, design, implementation and evaluation – can be iterated within themselves, and it is not uncommon to iterate between these steps (Figure 2.2). The important point, from a project management perspective, is to minimise the number of backwards steps; taking more than one step back can be very expensive in terms of time and resource. Above all, it is important to ensure that the overall direction of travel is towards delivery.

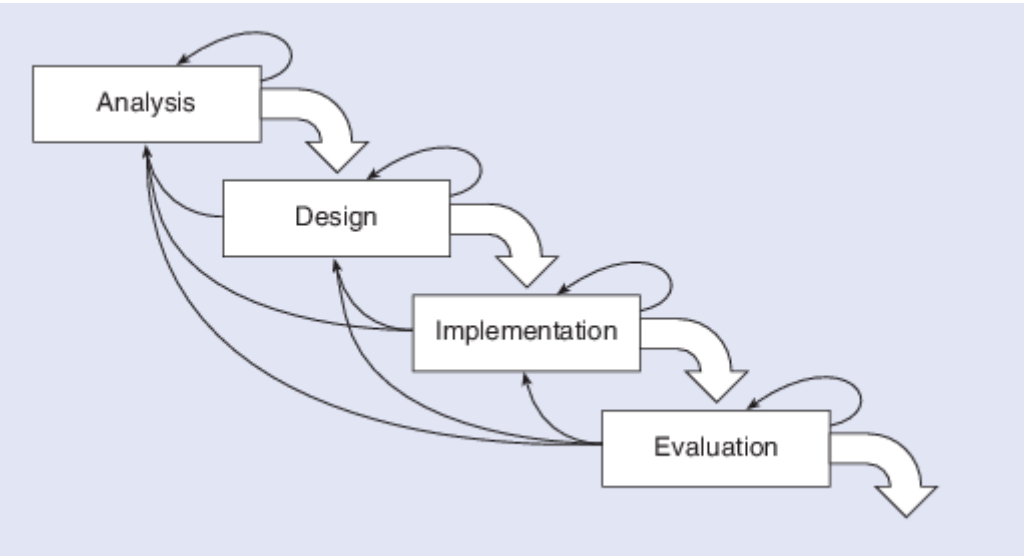

Figure 2.2 Iterative project lifecycle

Prototyping

A useful example of iterative development in this context is *prototyping*. Prototyping involves the development of an outline solution that is then taken back to those who have commissioned it (and those who will use or experience it) in order to confirm and clarify their requirements and facilitate an improved specification. It can be useful in giving users or clients an early appreciation of the capability, look and feel of the final system, and sparking off a dialogue about their requirements. Of course, that may result in a significant redefinition of the problem and its likely solution, and ideas about

how it can be implemented and evaluated. Prototypes, by their very nature, tend to encourage (or perhaps narrow) thinking along particular lines in respect of design, implementation and evaluation. It may also lock expectations into place, and tempt you to commit to the investment in the prototype rather than thinking through the design issues.

Prototyping has a particularly prominent role to play in projects that fall under the headings of human-computer interaction (HCI) and interaction design (IxD).

The star model

Hartson and Hix (1989) present the star model (Figure 2.3). Developed in the context of observed behaviour among HCI developers, it is very flexible in terms of what activities are performed when, but it does place evaluation as the central activity, implying that when you move from any one task to another you undertake an appropriate evaluation of where you are and where you are headed.

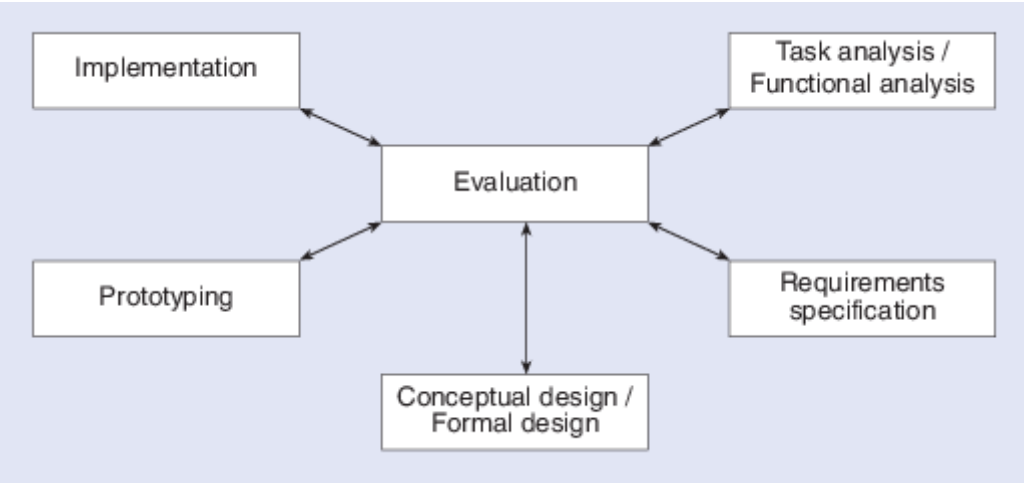

Figure 2.3 The star model

Interaction design model

Preece, Rogers and Sharp (2002) present a simple model for interaction design (Figure 2.4). It too is very unprescriptive. Evaluation has a more prominent role in the model than is suggested by the figure because, in moving between '(Re)design' and 'Identify needs/requirements', or between '(Re)design' and 'Build an interactive version', there would be some evaluative activity that indicated the need to move from one activity to another.

Figure 2.4 The interaction design (IxD) model

Structured-case model

A further example of an iterative lifecycle model is structured-case (Carroll and Swatman, 2000). The term 'case' is used to define the object of study; for instance, a person, a group, a project or programme, an organisation, a process, an information system, etc. The goal of structured-case is to produce new, revised or refined knowledge and theory that describe relationships between meaningful and important concepts and which are demonstrably rooted in observation. Structured-case may be distinguished from other forms of case study by the use of three steps in a process cycle that features:

- an evolving *conceptual framework* representing the current state of a researcher's aims, theoretical foundations and understandings. The researcher begins with an initial conceptual framework based on prior knowledge and experience, CF_1, iteratively revising it (CF_i) until their enquiry terminates, producing CF_n
- an iterative research cycle featuring data collection, analysis, (re)interpretation and synthesis
- ongoing literature-based scrutiny that is used to compare and contrast the evolving outcomes of the enquiry with what is known from the literature and which may either support or contest them.

The structured-case model is represented diagrammatically in Figure 2.5.

Figure 2.5 The structured-case model

Structured-case recognises that, for many projects, and especially ill-defined projects, the reality is that you start with an initial, often basic, understanding of a problem together with an idea or two about how to develop that understanding just a little more through observation, enquiry and/or experiment. You anticipate that having undertaken a little more enquiry or experiment, followed by some reflective analysis, you will end up with a better, deeper or more comprehensive understanding and that you will then be in a better position to see where to go next. This approach to structuring your project activity can be quite helpful:

- it recognises that you come to a problem with some initial preconceptions, which may or may not turn out to be well founded
- it gets away from the idea that you need to study some vast body of literature on the topic before you even start work
- it acknowledges that you can't always predetermine the attainment of the eventual outcome or goal and accepts that a successful project is one which makes a discernible and helpful contribution to everyone's understanding of a problem area.

Structured-case is particularly suitable for what might be called 'messy' projects where there are many stakeholders with different and often competing perspectives and interests, and a lack of certainty as to what the solution is or how a solution might be developed. In that sense, it helps to structure projects that are open-ended and exploratory.

2.3 Incrementation in project lifecycles

The subtle but important distinction between incremental and iterative development is that the goal of an *incremental* project is to develop an increasingly elaborate output, whereas an *iterative* model strives to develop an increasingly correct (less error-prone) output.

An incremental approach to project management may be adopted when there is some core product that provides a basic working solution and which can then be elaborated to provide more and richer features. At each stage, there is a working, albeit partial, solution to the problem. When adopting an incremental lifecycle model, it is vital to agree with whoever commissioned the project which features are most essential and which less essential.

It is also important to identify dependencies so that the work can be undertaken in the right sequence. In some cases there can be a strong dependence between the core product and the elaboration; in other cases they can be relatively independent. Analysing dependencies and integrating the additional features can be complex tasks in themselves.

It is important to recognise that it is possible to adopt an incremental approach where each incremental enhancement (including the first) is itself delivered using one of the lifecycle models considered so far.

2.4 Risk in project lifecycles

Risk assessment can be used to determine your approach to managing a project. By identifying risks you not only highlight potential uncertainties that need to be managed, but, done thoroughly and exhaustively, you also explore a range of alternative solutions and come to a much better understanding of the best way forward.

You have already seen how sequencing, iteration and increment can be applied in a lifecycle to improve project outputs. The same ideas can be applied to a lifecycle as a whole. Taking a step back to consider a solution in a broader context allows you to identify and manage the *risks* associated with a project. For instance:

- In a typical project, the major risks are around the requirements. Do you understand them? Have you got them all? Are they volatile?

- In many projects, especially research projects, there is often an element of the unknown about how to solve a problem, even once it is well defined.
- In any project there is likely to be a trade-off between what can be delivered in a given time and a specified set of available resources. Often, the number of desirable features of a solution exceeds those that can be delivered for a given cost and in a given time scale, so that choices have to be made.

One way to make such choices is to estimate the risk of not delivering each feature.

Boehm's spiral model

In a seminal paper on project management, Boehm (1988) proposed a spiral lifecycle model. There are many interpretations of the model in the literature. Figure 2.6 shows a version developed for this module, in which there are four steps that are repeated with each iteration of the spiral:

- assess the risks of all alternatives in the next cycle of activity and choose a course of action
- for the chosen course of action, identify objectives, plan and schedule next activities
- undertake the planned activities
- evaluate progress in this cycle.

Figure 2.6 A spiral lifecycle model

The spiral process starts, naturally enough, when it is recognised that there is a particular problem to address. The expanding cone of development in Figure 2.6 is intended to show growth in the scope of the developing solution and to reflect the resources that have been used, such as the cumulative cost of a project.

There is no need for a complete solution to be produced by the end of the first iteration of the spiral. For example, the first iteration could focus on the question (risk): 'Can we build an acceptable software system with the resources that can be brought to bear?' It would then identify the objectives, which might include finding out who the stakeholders were and evaluating their notions of 'acceptable'; it might include identifying functional requirements in more detail, assessing and costing the resources available. Activities that produced all this information would be planned and performed. The output would then be evaluated. The second cycle (assuming the first resulted in a decision to proceed) might aim to build a prototype. The activities for this would be planned and undertaken, the result evaluated. And so on.

After each cycle, some additional risks often come to light. With successive cycles, you should reach a point where your review indicates that you have reached an acceptable solution, or alternatively that you should stop!

Such a risk-driven model is particularly helpful when undertaking large projects or projects for which the developers are unfamiliar with the problem domain. In either case there is a higher risk of failure.

Development and risk

In taking a risk-driven approach to project management, it can be tempting to tackle the low-risk tasks before the high-risk tasks. However, in discussing incremental project lifecycles we met the ideas of dependency between project elements and of the relative importance of different elements to whoever commissioned the project. In these cases, analysis of dependency, importance and risk have significant implications for project management. If a project can be tackled incrementally, but the increments depend significantly on just one or two high-risk core elements, or if the high-value elements are also high-risk, then it becomes important to make sure you tackle the high-importance high-risk elements first.

2.5 The 'fractal' nature of lifecycle models

Your first response to this subheading is probably along the lines of ‘what does fractal mean?’ The basic idea behind fractals is *self-similarity*. Here is a common, if not entirely formal, illustration. Imagine viewing a large stretch of coastline from space. It is likely to be characterised by an uneven and ragged boundary. Now zoom in a little on a smaller stretch of that coastline; this too is uneven and ragged, with many headlands and bays. Keep zooming in, down to individual rocks, grains of sand, or even their microscopic surfaces. At each scale of magnification you will see an uneven and ragged boundary. Coastlines are like fractals because they are self-similar at every level of magnification.

Using this idea, we want you to think a little further about projects and lifecycles. Figure 2.7 presents a very common representation of the waterfall lifecycle for a routine software development project. Figure 2.8 does likewise in respect of database development projects.

Figure 2.7 Activities and outputs in the waterfall model for routine software development

Figure 2.8 Activities and outputs in the waterfall model for database development

For students of interaction design, Mayhew’s usability engineering lifecycle (see Preece, Rogers and Sharp, 2002, pp. 194–5) represents an interaction design project lifecycle at this same level of scale (or ‘magnification’).

Now, given a fairly large and complex project, if we zoom in on any of the subtasks, it too can be considered as a smaller ‘project within a project’. As such, it can undergo development through a series of phases involving the four activities of analysis, design, implementation and evaluation.

There is an important message for you here, as you think about the scope of your project. If your initial idea for a project involves a relatively large and complex system, you should think seriously about re-scoping the project. When you are identifying the subtasks that make up your project, and the sub-subtasks, choose a subtask that is still a substantive piece of work that needs to have requirements identified, a design developed and then implemented and evaluated. Such a subtask could be to deliver some software that supplies one or two elements of the functionality of a system, it might be a comprehensive and complex testing scheme needed to integrate a new piece of software into an existing large and complex legacy system, or it might be a scheme to evaluate alternative ‘off-the-shelf’ packages, where the criteria, evaluative measures and set of alternative packages are all to be determined prior to the evaluation.

The point of drawing your attention to this issue is that to pass this module you need to demonstrate your abilities in respect of the four main activities in any project (analysis, design, development and evaluation) and to manage those activities within a chosen project lifecycle model. This is achievable in many different ways and does not require you to tackle a large-scale problem involving the development of a whole computing system. If the scope of your project is too ambitious, you run the risk of running out of time without having progressed through all four activities at least once.

3 Choosing an approach to project management

In guiding you towards choosing a lifecycle model for your project we shall consider two questions: the nature of your proposed project and the manageability of the risks associated with it.

3.1 The nature of your project

The challenge is to choose, plan and manage your project in a way that takes into account its characteristics in terms of the strengths and weaknesses of the models we have considered (or any others you are familiar with). For some projects (but we suspect very few) the choice may be obvious, even at this stage. Nonetheless, we are looking for you to demonstrate that you have thoroughly considered at least one leading alternative.

It is not possible for us to consider the characteristics of your particular project in this guidance; instead, we will help you to identify the important characteristics of the models above and you can then consider these as selection criteria in relation to your proposed project.

Obvious strengths

We begin by recording the obvious strengths of each of the main models. In Table 3.1 we have begun to complete some of the entries. We leave it to you to complete or extend the table.

Table 3.1 Lifecycle models – main strengths

Model	Main strengths
Classic waterfall	Good when there is a very limited number of stakeholders, consensus about the solution, everyone gains, the task is small, can be evaluated comprehensively, familiar with just about all aspects of the problem and there is a record of past success, limited impact.
Iterative waterfall	
Prototyping (e.g. star and IxD)	
Incremental	Good when there is a core solution that meets a valuable if partial need and which can be gradually built on, subject to time and other resources.
Risk-driven	

Others …

Once you have a picture of the general strengths of each approach, you can then refine the table by describing the strengths of each model as they relate directly to your project.

Identifying relative strengths

Constructing Table 3.1 has given a number of criteria against which to consider your proposed project. We can extend this set by contrasting the models in a pair-wise fashion. By contrasting, rather than comparing, we can identify criteria that help us to choose one over the other.

Again, we devise a table to achieve this (Table 3.2). In this table, each cell records the main difference(s) between each pairing. Some cells may be easy to complete, others may be more difficult. It is helpful for you to complete all the cells, but if the entries become increasingly 'forced' or contrived, this is often a signal that they are less significant as selection criteria.

Table 3.2 Lifecycle models – main differences

	Classic waterfall	**Iterative waterfall**	**Structured-case**	**Incremental**
Classic waterfall	–	–	–	–
Iterative waterfall		–		–
Structured-case	The extent to which the goal and how to attain it are well defined and understood.		–	–
Incremental			The extent to which the next enhancement is well understood on its commencement.	–
Risk-driven	The extent to which all stages of the project are perceived to be 'doable' at its very outset.			

Just for completeness, we show you an extension to this technique for identifying differences between alternatives, which is good at unearthing the least obvious criteria (Table 3.3). This can be helpful where the criteria you have fleshed out so far are not sufficient to make a decision. The technique involves selecting three elements so that two are grouped together and then contrasted with the third. Then look for a difference that exists between the group of two and the third that is not also a difference between the two in the group.

Table 3.3 Lifecycle models – finding less obvious differences

	Structured-case
Risk-driven and **Classic waterfall**	In many situations where the structured-case approach is applicable, it is because the next iterative step is so exploratory that an analysis of the risks to success inherent in pursuing a particular path or line of enquiry is not easily assessable at any level.

We encourage you to try this approach, simply because it has been shown to be a useful analytical technique in other situations and it may support your decision making. However, there is no need to be exhaustive, not least because the differences can become very difficult to articulate and this may signal that they are not that important.

3.2 Risk assessment

Risk can be considered as arising both from ‘the significance or impact of things that might happen’ and ‘how likely they are to happen’. Although it is tempting to imagine that the likelihood of something happening can be numerically evaluated, in practice this is difficult or even impossible. It is better to be pragmatic and take a qualitative approach to risk assessment, in which impact and likelihood can take only restricted values. One common approach is to use a three-valued scale: low, medium or high. These values are best interpreted in relation to one another: for instance, a low-impact event will be less damaging than a medium-impact event. A medium-likelihood event will occur with less certainty than a high-likelihood event.

You should always aim to include some rough-and-ready definition of what ‘low’, ‘medium’ and ‘high’ are to be taken to mean. For example, definitions of levels of impact might be:

- low impact means ‘has negligible effect on the project, e.g. a delay of a few days’
- medium impact means ‘has considerable effect on the project, but it is still likely to succeed’
- high impact means ‘the project’s success is threatened’.

Definitions of likelihood might be:

- low likelihood means ‘it would be considered surprising if it occurred or proved to be the case’
- medium likelihood means ‘it would be disappointing but not surprising if it occurred’
- high likelihood means ‘will very probably prove to be the case’.

Using these scales, we can combine impact and likelihood to produce a risk combination table, which provides a measure of risk. If we consider impact and likelihood as being equally important, we obtain the following risk combination table (Table 3.4).

Table 3.4 Risk combination table

		Medium likelihood	High likelihood
Low impact	low risk	low risk	medium risk
Medium impact	low risk	medium risk	high risk
High impact	medium risk	high risk	high risk

If impact and likelihood are weighted differently, then the entries in the table will change. (Consider sketching a table where impact is given greater weight than likelihood.)

3.3 Risk management

Having identified and rated risks, you need to determine which ones to address, and how. Your decisions will be based on several interrelated factors, including the resources available for treating risks (medium risks that are easily addressed might be considered before high risks that are very difficult to address) and past experience of similar risks.

In determining which will be addressed, there are a number of possibilities. You may decide that:

- no risks are acceptable – all risks, whether low, medium or high, should eventually be addressed
- low risks are acceptable – only medium and high risks should be addressed
- low and medium risks are acceptable – only high risks should be addressed.

In determining how they might be addressed, you have a number of options. Conventionally, these are represented as:

- avoid the risk entirely by taking a course of action that eliminates it, including the possibility of undertaking a different project
- assign the task to another party in such a way as to reduce the risk to an acceptable level; for example, they may have more experience or better resources
- insure against the risk
- accept the risk.

In the very early stages of this module you have the opportunity to avoid very risky projects by choosing a different topic, or the same topic but addressing a different aspect of the problem. Once your project is underway, but still in the early stages, you may also avoid high risks by focusing on one aspect of a problem rather than another. Once your project is well underway, you may still have some room for manoeuvre, but, increasingly, you will need to accept the risks you identify. We certainly don’t recommend assigning your project to someone else – we think that is a risk with a very high impact and a very high likelihood of being detected!

4 Balancing project management and practical work

At this stage, you should have identified a number of aspects of the different lifecycle models that can be used as selection criteria against which you can assess your proposed project. For each of the criteria you consider to be the most important, you should be able to write a short statement (a couple of sentences or so) saying how it relates to your project and thus to choose one model in preference to others.

Finally, we want to make an important point in relation to the conduct of your project.

The ability to choose and apply a project management approach, and refining it as you proceed, is an important focus of assessment for this module. However, it is important to balance the project management aspects of what you do with making concrete practical progress in relation to solving the problem at the heart of your project. Both require you to develop and produce documentation, but to different ends. Knowing when to move from planning and management into doing, and when to step back from doing so that you can review, reflect and revise your plan is an important skill. We make this point by considering one of Escher's famous etchings (Figure 4.1). With all projects it is important to know when to stop musing about your project management problems (the man in the picture) and do something that makes concrete progress (the woman in the picture).

Figure 4.1 *Waterfall*, lithograph by M.C. Escher, 1961, http://www.mcescher.com/ (reproduced with permission)

5 References

Boehm, B.W. (1988) 'A spiral model of software development and enhancement', *Computer*, vol. 21, no. 5, pp. 61–72.

Carroll, J.M. and Swatman, P.A. (2000) 'Structured-case: a methodological framework for building theory in information systems research', *European Journal of Information Systems*, vol. 9, no. 8, pp. 235–42.

Hartson, H.R. and Hix, D. (1989) 'Towards empirically derived methodologies and tools for human-computer interface development', *International Journal of Man-Machine Studies*, vol. 31, pp. 477–94.

Preece, J., Rogers, Y. and Sharp, H. (2002) *Interaction Design*, John Wiley & Sons, Inc., New York.

Royce, W.W. (1970) 'Managing the development of large software systems', *Proceedings of the IEEE WESCON*, vol. 26, pp. 1–9.

6 Acknowledgements

The module team wishes to acknowledge the contributions of the following in producing this guidance:

Steve Armstrong

Jon Hall

Kevin Waugh.

Grateful acknowledgement is also made to the following for permission to reproduce the lithograph in Figure 4.1: © 2007 The M.C. Escher Company, Holland. All rights reserved. www.mcescher.com.

< Choosing a Lifecycle Model