Chapter 3

Project evaluation

OBJECTIVES

When you have completed this chapter you will be able to:

- carry out an evaluation and selection of projects against strategic, technical and economic criteria;
- use a variety of cost-benefit evaluation techniques for choosing among competing project proposals;
- evaluate the risk involved in a project and select appropriate strategies for minimizing potential costs.

3.1 Introduction

Deciding whether or not to go ahead with a project is really a case of comparing a proposed project with the alternatives and deciding whether to proceed with it. That evaluation will be based on strategic, technical and economic criteria and will normally be undertaken as part of strategic planning or a feasibility study for any information system development. The risks involved also need to be evaluated.

'Do nothing' is an option which should always be considered.

In this chapter we shall be using the term *project* in a broader sense than elsewhere in the book. Our decision as to whether or not to proceed with a project needs to be based upon whether or not it is desirable to carry out the development *and operation* of a software system. The term project may therefore be used, in this context, to describe the whole life cycle of a system from conception through to final decommissioning.

The BS 6079 guidelines (see Appendix B) use the term project in this way.

Project evaluation is normally carried out in Step 0 of Step Wise (Figure 3.1). The subsequent steps of Step Wise are then concerned with managing the development project that stems from this project selection.

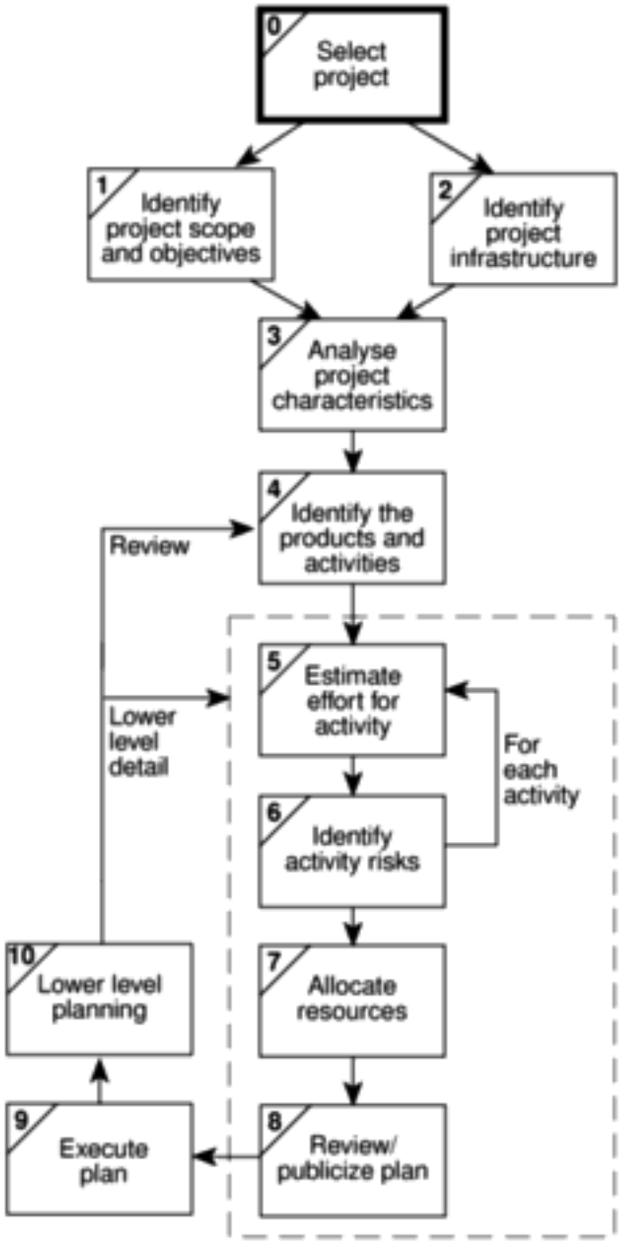


Figure 3.1 *Project evaluation is carried out in Step 0.*

D. C. Ferns defined a programme as 'a group of projects that are managed in a co-ordinated way to gain benefits that would not be possible were the projects to be managed independently' in a seminal article in the *International Journal of Project Management*, August 1991.

3.2 Strategic assessment

Programme management

It is being increasingly recognized that individual projects need to be seen as components of a *programme* and should be evaluated and managed as such. A programme, in this context, is a collection of projects that all contribute to the same overall organizational goals. Effective programme management requires that there is a well defined *programme goal* and that all the organization's projects are selected and tuned to contribute to this goal. A project must be evaluated according to how it contributes to this programme goal and its viability, timing, resourcing and final worth can be affected by the programme as a whole. It is to be expected

that the value of any project is increased by the fact that it is part of a programme – the whole, as they say, being greater than the sum of the parts.

In order to carry out a successful strategic assessment of a potential project there should therefore be a strategic plan clearly defining the organization's objectives. This provides the context for defining the programme and programme goals and, hence, the context for assessing the individual project. It is likely, particularly in a large organization, that there will be an organizational structure for programme management and it will be, for example, the *programme director* and *programme executive*, rather than, say, a project manager, who will be responsible for the strategic assessment of a proposed project.

Even where there is no explicitly defined programme, any proposed project must be evaluated within the context of the organization's overall business objectives. Moreover, any potential software system will form part of the user organization's overall information system and must be evaluated within the context of the existing information system and the organization's information strategy. Table 3.1 illustrates typical issues that must be addressed as part of the strategic assessment of a project.

Table 3.1 *Typical issues and questions to be considered during strategic assessment*

Issue	*Typical questions*
Objectives	How will the proposed system contribute to the organization's stated objectives? How, for example, might it contribute to an increase in market share?
IS plan	How does the proposed system fit into the IS plan? Which existing system(s) will it replace/interface with? How will it interact with systems proposed for later development?
Organization structure	What effect will the new system have on the existing departmental and organization structure? Will, for example, a new sales order processing system overlap existing sales and stock control functions?
MIS	What information will the system provide and at what levels in the organization? In what ways will it complement or enhance existing management information systems?
Personnel	In what way will the proposed system affect manning levels and the existing employee skill base? What are the implications for the organization's overall policy on staff development?
Image	What, if any, will be the effect on customers' attitudes towards the organization? Will the adoption of, say, automated systems conflict with the objectives of providing a friendly service?

Where a well-defined information systems strategy does not exist, system development and the assessment of project proposals will be based on a more piecemeal approach – each project being individually assessed early in its life cycle. In such cases it is likely that cost–benefit analysis will have more importance and some of the questions of Table 3.1 will be more difficult to answer.

Portfolio management

Third party developers must also carry out strategic and operational assessment of project proposals.

Where an organization such as a software house is developing a software system they could be asked to carry out a strategic and operational assessment on behalf of the customer. Whether or not this should be the case, they will require an assessment of any proposed project themselves. They will need to ensure that carrying out the development of a system is consistent with their own strategic plan – it is unlikely, for example, that a software house specializing in financial and accounting systems would wish to undertake development of a factory control system unless their strategic plan placed an emphasis on diversification.

The proposed project will form part of a *portfolio* of ongoing and planned projects and the selection of projects must take account of the possible effects on other projects in the portfolio (competition for resources, for example) and the overall portfolio profile (for example, specialization versus diversification).

3.3 Technical assessment

Technical assessment of a proposed system consists of evaluating the required functionality against the hardware and software available. Where an organization has a strategic information systems plan, this is likely to place limitations on the nature of solutions that might be considered. The constraints will, of course, influence the cost of the solution and this must be taken into account in the cost–benefit analysis.

3.4 Cost–benefit analysis

The most common way of carrying out an economic assessment of a proposed information system, or other development, is by comparing the expected costs of development and operation of the system with the benefits of having it in place.

Any project requiring an investment must, as a minimum, provide a greater benefit than putting that investment in, say, a bank.

Assessment is based upon the question of whether the estimated costs are exceeded by the estimated income and other benefits. Additionally, it is usually necessary to ask whether or not the project under consideration is the best of a number of options. There might be more candidate projects than can be undertaken at any one time and, in any case, projects will need to be prioritized so that any scarce resources may be allocated effectively.

The standard way of evaluating the economic benefits of any project is to carry out a cost–benefit analysis, which consists of two steps.

- **Identifying and estimating all of the costs and benefits of carrying out the project** This includes development costs of the system, the operating costs and the benefits that are expected to accrue from the operation of the system. Where the proposed system is replacing an existing one, these estimates should reflect the costs and benefits due to the new system. A sales order processing system, for example, could not claim to benefit an organization by the total value of sales – only by the increase due to the use of the new system.
- **Expressing these costs and benefits in common units** We must evaluate the net benefit, which is the difference between the total benefit and the total cost. To do this, we must express each cost and each benefit in monetary terms.

Most costs are relatively easy to identify and quantify in approximate monetary terms. It is helpful to categorize costs according to where they originate in the life of the project.

Many costs are easy to identify and measure in monetary terms.

- **Development costs** – include the salaries and other employment costs of the staff involved in the development project and all associated costs.
- **Setup costs** – include the costs of putting the system into place. These consist mainly of the costs of any new hardware and ancillary equipment but will also include costs of file conversion, recruitment and staff training.
- **Operational costs** – consist of the costs of operating the system once it has been installed.

Benefits, on the other hand, are often quite difficult to quantify in monetary terms even once they have been identified. Benefits may be categorized as follows.

Indirect benefits, which are difficult to estimate, are sometimes known as intangible benefits.

- **Direct benefits** – these accrue directly from the operation of the proposed system. These could, for example, include the reduction in salary bills through the introduction of a new, computerized system.
- **Assessable indirect benefits** – these are generally secondary benefits, such as increased accuracy through the introduction of a more user-friendly screen design where we might be able to estimate the reduction in errors, and hence costs, of the proposed system.
- **Intangible benefits** – these are generally longer term or benefits that are considered very difficult to quantify. Enhanced job interest can lead to reduced staff turnover and, hence, lower recruitment costs.

Exercise 3.1

Brightmouth College are considering the replacement of the existing payroll service, operated by a third party, with a tailored, off-the-shelf computer-based system. List some of the costs and benefits they might consider under each of the six headings given above. For each cost or benefit, explain how, in principle, it might be measured in monetary terms.

If a proposal shows an excess of benefits over costs then it is a candidate for further consideration.

Any project that shows an excess of benefits over costs is clearly worth considering for implementation. However, as we shall see later, it is not a sufficient justification for going ahead: we might not be able to afford the costs; there might be even better projects we could allocate our resources to instead; the project might be too risky.

3.5 Cash flow forecasting

As important as estimating the overall costs and benefits of a project is the forecasting of the cash flows that will take place and their timing. A cash flow forecast will indicate when expenditure and income will take place (Figure 3.2).

Typically products generate a negative cash flow during their development followed by a positive cash flow over their operating life. There might be decommissioning costs at the end of a product's life

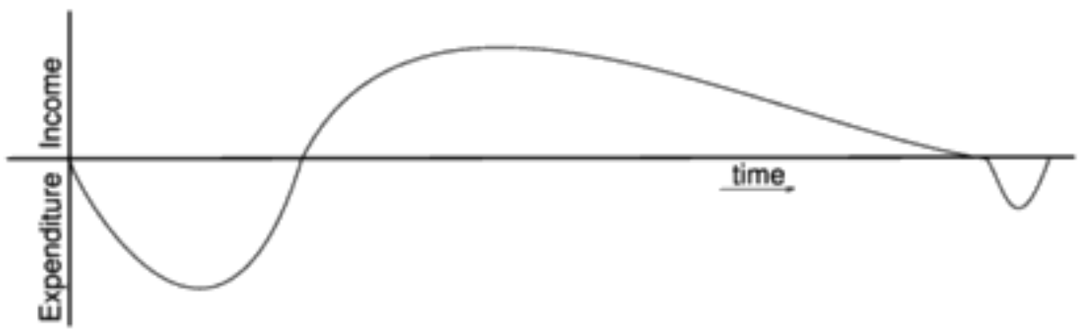


Figure 3.2 *Typical product life cycle cash flow.*

We need to spend money, such as staff wages, during the development stages of a project. Such expenditure cannot be deferred until income is received (either from using the software if it is being developed for in-house use or from selling it). It is important that we know that we can fund the development expenditure either from the company's own resources or by borrowing from the bank. In any event, it is vital to have some forecast of when expenditure such as the payment of salaries and bank interest will take place and when any income is to be expected, such as payment on completion or, possibly, stage payments.

The difficulty and importance of cash flow forecasting is evidenced by the number of companies that suffer bankruptcy because, although they are developing profitable products or services, they cannot sustain an unplanned negative cash flow.

Accurate cash flow forecasting is not easy, as it generally needs to be done early in the project's life cycle (at least before any significant expenditure is committed) and many items to be estimated (particularly the benefits of using software or decommissioning costs) might be some years in the future.

When estimating future cash flows, it is usual to ignore the effects of inflation. Trying to forecast the effects of inflation increases the uncertainty of the forecasts. Moreover, if expenditure is increased due to inflation it is likely that income will increase proportionately. However, measures to deal with increases in costs where work is being done for an external customer must be in place – for example index linked prices where work involves use of raw materials – see Chapter 10 on contract management.

Table 3.2 illustrates cash flow forecasts for four projects. In each case it is assumed that the cash flows take place at the end of each year. For short-term projects or where candidate projects demonstrate significant seasonal cash flow

patterns it can be advisable to produce quarterly, or even monthly, cash flow forecasts.

3.6 Cost–benefit evaluation techniques

We would consider proceeding with a project only where the benefits outweigh the costs. However, in order to choose among projects, we need to take into account the timing of the costs and benefits as well as the benefits relative to the size of the investment.

Exercise 3.2

Consider the project cash flow estimates for four projects at IOE shown in Table 3.2. Negative values represent expenditure and positive values income.

Rank the four projects in order of financial desirability and make a note of your reasons for ranking them in that way before reading further.

In the following sections we will take a brief look at some common methods for comparing projects on the basis of their cash flow forecasts.

Net profit

The net profit of a project is the difference between the total costs and the total income over the life of the project. Project 2 in Table 3.2 shows the greatest net profit but this is at the expense of a large investment. Indeed, if we had £1m to invest, we might undertake all of the other three projects and obtain an even greater net profit. Note also, that all projects contain an element of risk and we might not be prepared to risk £1m. We shall look at the effects of risk and investment later in this chapter.

Table 3.2 *Four project cash flow projections – figures are end of year totals (£)*

Year	*Project 1*	*Project 2*	*Project 3*	*Project 4*
0	−100,000	−1,000,000	−100,000	−120,000
1	10,000	200,000	30,000	30,000
2	10,000	200,000	30,000	30,000
3	10,000	200,000	30,000	30,000
4	20,000	200,000	30,000	30,000
5	100,000	300,000	30,000	75,000
Net profit	50,000	100,000	50,000	75,000

Cash flows take place at the end of each year. The year 0 figure represents the initial investment made at the start of the project.

Moreover, the simple net profit takes no account of the timing of the cash flows. Projects 1 and 3 each have a net profit of £50,000 and therefore, according to this selection criterion, would be equally preferable. The bulk of the income occurs late in the life of project 1, whereas project 3 returns a steady income throughout

its life. Having to wait for a return has the disadvantage that the investment must be funded for longer. Add to that the fact that, other things being equal, estimates in the more distant future are less reliable that short-term estimates and we can see that the two projects are not equally preferable.

Payback period

The *payback period* is the time taken to break even or pay back the initial investment. Normally, the project with the shortest payback period will be chosen on the basis that an organization will wish to minimize the time that a project is 'in debt'.

Exercise 3.3

Consider the four project cash flows given in Table 3.2 and calculate the payback period for each of them.

The advantage of the payback period is that it is simple to calculate and is not particularly sensitive to small forecasting errors. Its disadvantage as a selection technique is that it ignores the overall profitability of the project – in fact, it totally ignores any income (or expenditure) once the project has broken even. Thus the fact that projects 2 and 4 are, overall, more profitable than project 3 is ignored.

Return on investment

The *return on investment* (ROI), also known as the *accounting rate of return* (ARR), provides a way of comparing the net profitability to the investment required. There are some variations on the formula used to calculate the return on investment but a straightforward common version is

$$\text{ROI} = \frac{\text{average annual profit}}{\text{total investment}} \times 100$$

Exercise 3.4

Calculating the ROI for project 1, the net profit is £50,000 and the total investment is £100,000. The return on investment is therefore calculated as

$$\begin{aligned}\text{ROI} &= \frac{\text{average annual profit}}{\text{total investment}} \times 100 \\ &= \frac{10{,}000}{100{,}000} \times 100 = 10\%\end{aligned}$$

Calculate the ROI for each of the other projects shown in Table 3.2 and decide which, on the basis of this criterion, is the most worthwhile.

The return on investment provides a simple, easy to calculate measure of return on capital and is therefore quite popular. Unfortunately it suffers from two severe disadvantages. Like the net profitability, it takes no account of the timing of the cash flows. More importantly, it is tempting to compare the rate of return with current interest rates. However, this rate of return bears no relationship to the interest rates offered or charged by banks (or any other normal interest rate) since it takes no account of the timing of the cash flows or of the compounding of interest. It is therefore, potentially, very misleading.

Net present value

Net present value (NPV) and internal rate of return (IRR) are collectively known as discounted cash flow (DCF) techniques.

The calculation of *net present value* is a project evaluation technique that takes into account the profitability of a project and the timing of the cash flows that are produced. It does so by discounting future cash flows by a percentage known as the discount rate. This is based on the view that receiving £100 today is better than having to wait until next year to receive it, because the £100 next year is worth less than £100 now. We could, for example, invest the £100 in a bank today and have £100 plus the interest in a year's time. If we say that the present value of £100 in a year's time is £91, we mean that £100 in a year's time is the equivalent of £91 now.

Note that this example uses approximate figures – when you have finished reading this section you should be able to calculate the exact figures yourself.

The equivalence of £91 now and £100 in a year's time means we are discounting the future income by approximately 10% – that is, we would need an extra 10% to make it worthwhile waiting for a year. An alternative way of considering the equivalence of the two is to consider that, if we received £91 now and invested for a year at an annual interest rate of 10%, it would be worth £100 in a year's time. The annual rate by which we discount future earnings is known as the *discount rate* – 10% in the above example.

Similarly, £100 received in 2 years' time would have a present value of approximately £83 – in other words, £83 invested at an interest rate of 10% would yield approximately £100 in 2 years' time.

The present value of any future cash flow may be obtained by applying the following formula

$$\text{present value} = \frac{\text{value in year } t}{(1+r)^t}$$

where r is the discount rate, expressed as a decimal value and t is the number of years into the future that the cash flow occurs.

Alternatively, and rather more easily, the present value of a cash flow may be calculated by multiplying the cash flow by the appropriate discount factor. A small table of discount factors is given in Table 3.3.

The NPV for a project is obtained by discounting each cash flow (both negative and positive) and summing the discounted values. It is normally assumed that any initial investment takes place immediately (indicated as year 0) and is not discounted. Later cash flows are normally assumed to take place at the end of each year and are discounted by the appropriate amount.

Table 3.3 *Table of NPV discount factors*

	Discount rate (%)					
Year	*5*	*6*	*8*	*10*	*12*	*15*
1	0.9524	0.9434	0.9259	0.9091	0.8929	0.8696
2	0.9070	0.8900	0.8573	0.8264	0.7972	0.7561
3	0.8638	0.8396	0.7938	0.7513	0.7118	0.6575
4	0.8227	0.7921	0.7350	0.6830	0.6355	0.5718
5	0.7835	0.7473	0.6806	0.6209	0.5674	0.4972
6	0.7462	0.7050	0.6302	0.5645	0.5066	0.4323
7	0.7107	0.6651	0.5835	0.5132	0.4523	0.3759
8	0.6768	0.6274	0.5403	0.4665	0.4039	0.3269
9	0.6446	0.5919	0.5002	0.4241	0.3606	0.2843
10	0.6139	0.5584	0.4632	0.3855	0.3220	0.2472
15	0.4810	0.4173	0.3152	0.2394	0.1827	0.1229
20	0.3769	0.3118	0.2145	0.1486	0.1037	0.0611
25	0.2953	0.2330	0.1460	0.0923	0.0588	0.0304

More extensive or detailed tables may be constructed using the formula

$$\text{discount factor} = \frac{1}{(1+r)^t}$$

for various values of r (the discount rate) and t (the number of years from now)

Exercise 3.5

Assuming a 10% discount rate, the NPV for project 1 (Table 3.2) would be calculated as in Table 3.4. The net present value for Project 1, using a 10% discount rate is therefore £618. Using a 10% discount rate, calculate the net present values for projects 2, 3 and 4 and decide which, on the basis of this, is the most beneficial to pursue.

Table 3.4 *Applying the discount factors to project 1*

Year	*Project 1 cash flow (£)*	*Discount factor @ 10%*	*Discounted cash flow (£)*
0	−100,000	1.0000	−100,000
1	10,000	0.9091	9,091
2	10,000	0.8264	8,264
3	10,000	0.7513	7,513
4	20,000	0.6830	13,660
5	100,000	0.6209	62,090
Net Profit:	£50,000		NPV: £618

It is interesting to note that the net present values for projects 1 and 3 are significantly different – even though they both yield the same net profit and both have the same return on investment. The difference in NPV reflects the fact that, with project 1, we must wait longer for the bulk of the income.

The main difficulty with NPV for deciding between projects is selecting an appropriate discount rate. Some organizations have a standard rate but, where this is not the case, then the discount rate should be chosen to reflect available interest rates (borrowing costs where the project must be funded from loans) plus some premium to reflect the fact that software projects are inherently more risky than lending money to a bank. The exact discount rate is normally less important than ensuring that the same discount rate is used for all projects being compared. However, it is important to check that the ranking of projects is not sensitive to small changes in the discount rate – have a look at the following exercise.

Exercise 3.6

Calculate the net present value for each of the projects A, B and C shown in Table 3.5 using each of the discount rates 8%, 10% and 12%.

For each of the discount rates, decide which is the best project. What can you conclude from these results?

Alternatively, the discount rate can be thought of as a target rate of return. If, for example, we set a target rate of return of 15% we would reject any project that did not display a positive net present value using a 15% discount rate. Any project that displayed a positive NPV would be considered for selection – perhaps by using an additional set of criteria where candidate projects were competing for resources.

Table 3.5 *Three estimated project cash flows*

Year	*Project A (£)*	*Project B (£)*	*Project C (£)*
0	– 8,000	– 8,000	– 10,000
1	4,000	1,000	2,000
2	4,000	2,000	2,000
3	2,000	4,000	6,000
4	1,000	3,000	2,000
5	500	9,000	2,000
6	500	–6,000	2,000
Net Profit	4,000	5,000	6,000

Internal rate of return

One disadvantage of NPV as a measure of profitability is that, although it may be used to compare projects, it might not be directly comparable with earnings from other investments or the costs of borrowing capital. Such costs are usually quoted

as a percentage interest rate. The internal rate of return (IRR) attempts to provide a profitability measure as a percentage return that is directly comparable with interest rates. Thus, a project that showed an estimated IRR of 10% would be worthwhile if the capital could be borrowed for less than 10% or if the capital could not be invested elsewhere for a return greater than 10%.

The IRR is calculated as that percentage discount rate that would produce an NPV of zero. It is most easily calculated using a spreadsheet or other computer program that provides functions for calculating the IRR. Microsoft Excel and Lotus, for example, both provide IRR functions which, provided with an initial guess or seed value (which may be zero), will search for and return an IRR.

Manually, it must be calculated by trial-and-error or estimated using the technique illustrated in Figure 3.3. This technique consists of guessing two values

The IRR may be estimated by plotting a series of guesses: For a particular project, a discount rate of 8% gives a positive NPV of £7,898; a discount rate of 12% gives a negative NPV of −£5,829. The IRR is therefore somewhere between these two values. Plotting the two values on a chart and joining the points with a straight line suggests that the IRR is about 10.25%. The true IRR (calculated with a spreadsheet) is 10.167%.

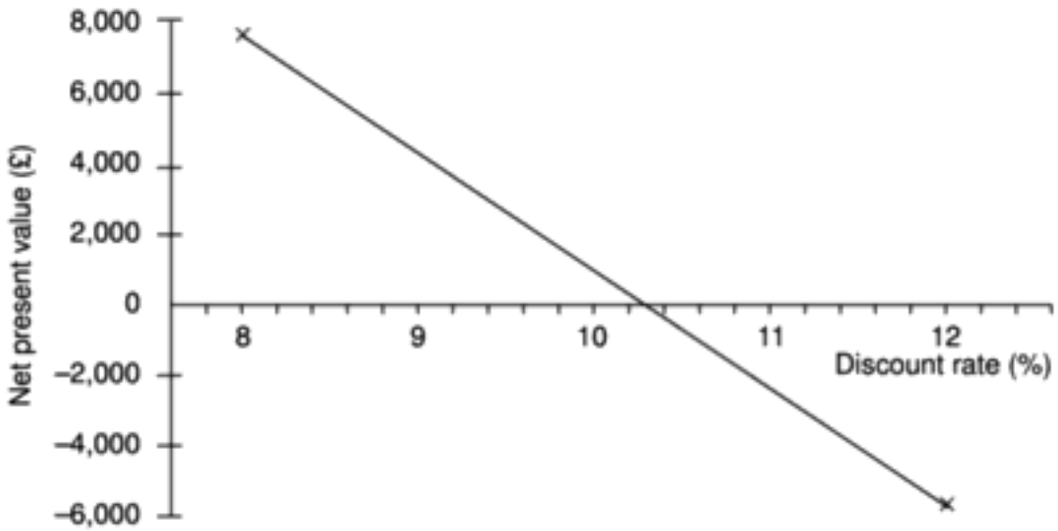


Figure 3.3 *Estimating the internal rate of return for project 1.*

(one either side of the true value) and using the resulting NPVs (one of which must be positive and the other negative) to estimate the correct value. Note that this technique will provide only an approximate value but, in many cases that can be sufficient to dismiss a project that has a small IRR or indicate that it is worth making a more precise evaluation.

The internal rate of return is a convenient and useful measure of the value of a project in that it is a single percentage figure that may be directly compared with rates of return on other projects or interest rates quoted elsewhere.

Table 3.6 illustrates the way in which a project with an IRR of 10% may be directly compared with other interest rates. The cash flow for the project is shown in column (a). Columns (b) to (e) show that if we were to invest £100,000 now at an annual interest rate of 10% in, say, a bank, we could withdraw the same amounts as we would earn from the project at the end of each year, column (e), and we would be left with a net balance of zero at the end. In other words, investing in a project that has an IRR of 10% can produce exactly the same cash flow as lending the money to a bank at a 10% interest rate. We can therefore reason

that a project with an IRR greater than current interest rates will provide a better rate of return than lending the investment to a bank. We can also say that it will be worth borrowing to finance the project if it has an IRR greater than the interest rate charged on the loan.

Table 3.6 *A project cash flow treated as an investment at 10%*

		Equivalent investment at 10%			
Year	*(a) Project cash flow forecast (£)*	*(b) Capital at start of year (£)*	*(c) Interest during year (£)*	*(d) Capital at end of year (£)*	*(e) End of year withdrawal (£)*
0	−100,000	—	—	—	—
1	10,000	100,000	10,000	110,000	10,000
2	10,000	100,000	10,000	110,000	10,000
3	10,000	100,000	10,000	110,000	10,000
4	20,000	100,000	10,000	110,000	20,000
5	99,000	90,000	9,000	99,000	99,000
6		0	0	0	0

£100,000 invested at 10% may be used to generate the cash flows shown. At the end of the 5-year period the capital and the interest payments will be entirely consumed leaving a net balance of zero.

One deficiency of the IRR is that it does not indicate the absolute size of the return. A project with an NPV of £100,000 and an IRR of 15% can be more attractive than one with an NPV of £10,000 and an IRR of 18% – the return on capital is lower but the net benefits greater.

An often quoted objection to the internal rate of return is that, under certain conditions, it is possible to find more than one rate that will produce a zero NPV. This is not a valid objection since, if there are multiple solutions, it is always appropriate to take the lowest value and ignore the others. Spreadsheets will normally always return the lowest value if provided with zero as a seed value.

NPV and IRR are not, however, a complete answer to economic project evaluation.

- A total evaluation must also take into account the problems of funding the cash flows – will we, for example, be able to repay the interest on any borrowed money and pay development staff salaries at the appropriate time?
- While a project's IRR might indicate a profitable project, future earnings from a project might be far less reliable than earnings from, say, investing with a bank. To take account of the risk inherent in investing in a project, we might require that a project earn a 'risk premium' (that is, it must earn, say, at least 15% more than current interest rates) or we might undertake a more detailed risk analysis as described in the following sections of this chapter.
- We must also consider any one project within the financial and economic framework of the organization as a whole – if we fund this one, will we also be able to fund other worthy projects?

3.7 Risk evaluation

There is a risk that software might exceed the original specification and that a project will be completed early and under budget. That is not a risk that need concern us.

Every project involves risk of some form. When assessing and planning a project, we are concerned with the risk of the project's not meeting its objectives. In Chapter 8 we shall discuss ways of analysing and minimizing risk during the development of a software system. In this chapter, we are concerned with taking risk into account when deciding whether to proceed with a proposed project.

Risk identification and ranking

In any project evaluation we should attempt to identify the risks and quantify their potential effects. One common approach to risk analysis is to construct a project risk matrix utilizing a checklist of possible risks and to classify each risk according to its relative importance and likelihood. Note that the importance and likelihood need to be separately assessed – we might be less concerned with something that, although serious, is very unlikely to occur than with something less serious that is almost certain. Table 3.7 illustrates a basic project risk matrix listing some of the risks that might be considered for a project, with their importance and likelihood classified as high (H), medium (M), low (L) or exceedingly unlikely (—). So that projects may be compared the list of risks must be the same for each project being assessed. It is likely, in reality, that it would be somewhat longer than shown and more precisely defined.

The project risk matrix may be used as a way of evaluating projects (those with high risks being less favoured) or as a means of identifying and ranking the risks for a specific project. In Chapter 7 we shall consider a method for scoring the importance and likelihood of risks that may be used in conjunction with the project risk matrix to score and rank projects.

Risk and net present value

Where a project is relatively risky it is common practice to use a higher discount rate to calculate net present value. This addition or risk premium, might, for example, be an additional 2% for a reasonably safe project or 5% for a fairly risky one. Projects may be categorized as high, medium or low risk using a scoring method and risk premiums designated for each category. The premiums, even if arbitrary, provide a consistent method of taking risk into account.

Cost–benefit analysis

A rather more sophisticated approach to the evaluation of risk is to consider each possible outcome and estimate the probability of its occurring and the corresponding value of the outcome. Rather than a single cash flow forecast for a project, we will then have a set of cash flow forecasts, each with an associated probability of occurring. The value of the project is then obtained by summing the cost or benefit for each possible outcome weighted by its corresponding probability. Exercise 3.7 illustrates how this may be done.

Table 3.7 *A fragment of a basic project risk matrix*

Risk	*Importance*	*Likelihood*
Software never completed or delivered	H	—
Project cancelled after design stage	H	—
Software delivered late	M	M
Development budget exceeded ≤ 20%	L	M
Development budget exceeded > 20%	M	L
Maintenance costs higher than estimated	L	L
Response time targets not met	L	H

Exercise 3.7

BuyRight, a software house, is considering developing a payroll application for use in academic institutions and is currently engaged in a cost-benefit analysis. Study of the market has shown that, if they can target it efficiently and no competing products become available, they will obtain a high level of sales generating an annual income of £800,000. They estimate that there is a 1 in 10 chance of this happening. However, a competitor might launch a competing application before their own launch date and then sales might generate only £100,000 per year. They estimate that there is a 30% chance of this happening. The most likely outcome, they believe, is somewhere in between these two extremes – they will gain a market lead by launching before any competing product becomes available and achieve an annual income of £650,000. BuyRight have therefore calculated their expected sales income as in Table 3.8.

Total development costs are estimated at £750,000 and sales are expected to be maintained at a reasonably constant level for at least four years. Annual costs of marketing and product maintenance are estimated at £200,000, irrespective of the market share gained. Would you advise them to go ahead with the project?

This approach is frequently used in the evaluation of large projects such as the building of new motorways, where variables such as future traffic volumes, and hence the total benefit of shorter journey times, are subject to uncertainty. The technique does, of course, rely on our being able to assign probabilities of occurrence to each scenario and, without extensive study, this can be difficult.

When used to evaluate a single project, the cost–benefit approach, by 'averaging out' the effects of the different scenarios, does not take account an organization's reluctance to risk damaging outcomes. Because of this, where overall profitability is the primary concern, it is often considered more appropriate for the evaluation of a portfolio of projects.

Risk profile analysis

An approach which attempts to overcome some of the objections to cost–benefit averaging is the construction of risk profiles using sensitivity analysis.

Table 3.8 *BuyRight's income forecasts*

Sales	*Annual sales income (£)* i	*Probability* p	*Expected Value (£)* $i \times p$
High	800,000	0.1	80,000
Medium	650,000	0.6	390,000
Low	100,000	0.3	30,000
Expected Income			500,000

This involves varying each of the parameters that affect the project's cost or benefits to ascertain how sensitive the project's profitability is to each factor. We might, for example, vary one of our original estimates by plus or minus 5% and recalculate the expected costs and benefits for the project. By repeating this exercise for each of our estimates in turn we can evaluate the sensitivity of the project to each factor.

By studying the results of a sensitivity analysis we can identify those factors that are most important to the success of the project. We then need to decide whether we can exercise greater control over them or otherwise mitigate their effects. If neither is the case, then we must live with the risk or abandon the project.

For an explanation of the Monte Carlo technique see any textbook on operational research.

Sensitivity analysis demands that we vary each factor one at a time. It does not easily allow us to consider the effects of combinations of circumstances, neither does it evaluate the chances of a particular outcome occurring. In order to do this we need to use a more sophisticated tool such as Monte Carlo simulation. There are a number of risk analysis applications available (such as *@Risk* from Palisade) that use Monte Carlo simulation and produce risk profiles of the type shown in Figure 3.4.

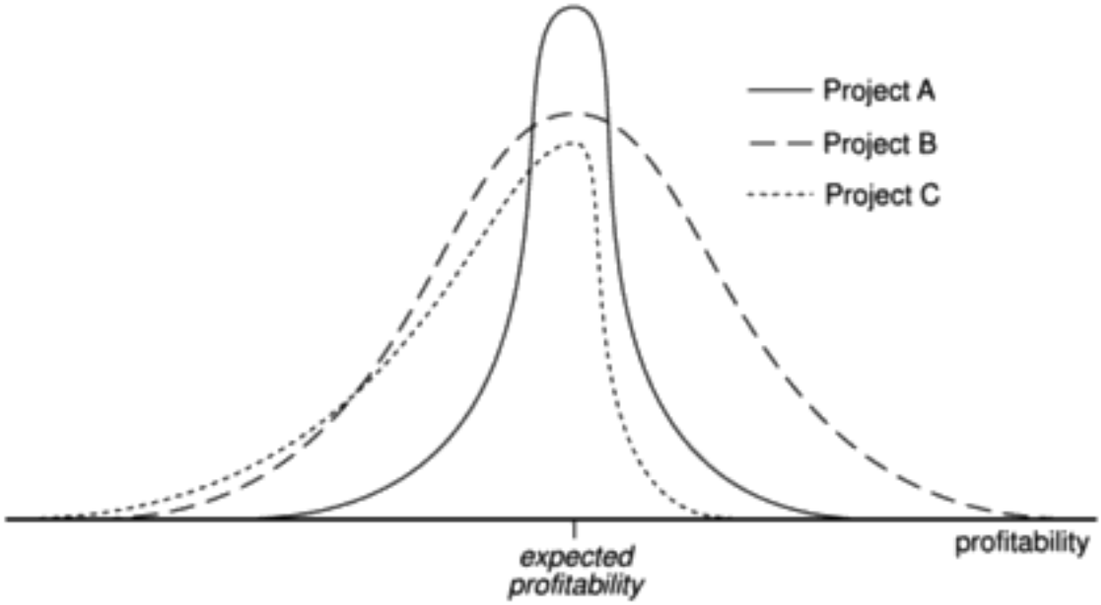
Projects may be compared as in Figure 3.4, which compares three projects with the same expected profitability. Project A is unlikely to depart far from this expected value compared to project B, which exhibits a larger variance. Both of these have symmetric profiles, which contrast with project C. Project C has a skewed distribution, which indicates that although it is unlikely ever to be much more profitable than expected, it is quite likely to be far worse.

Using decision trees

The approaches to risk analysis discussed previously rather assume that we are passive bystanders allowing nature to take its own course – the best we can do is to reject over-risky projects or choose those with the best risk profile. There are many situations, however, where we can evaluate whether a risk is important and, if it is, indicate a suitable course of action.

Many such decisions will limit or affect future options and, at any point, it is important to be able to see into the future to assess how a decision will affect the future profitability of the project.

Prior to giving Amanda the job of extending their invoicing system, IOE must consider the alternative of completely replacing the existing system – which they



All three projects have the same expected profitability.
The profitability of project A is unlikely to depart greatly from its expected value (indicated by the vertical axis) compared to the likely variations for project B. Project A is therefore less risky than project B.

Figure 3.4 *A risk analysis profile.*

will have to do at some point in the future. The decision largely rests upon the rate at which their equipment maintenance business expands – if their market share significantly increases (which they believe will happen if rumours of a competitor's imminent bankruptcy are fulfilled) the existing system might need to be replaced within 2 years. Not replacing the system in time could be an expensive option as it could lead to lost revenue if they cannot cope with the increase in invoicing demand. Replacing it immediately will, however, be expensive as it will mean deferring other projects that have already been scheduled.

They have calculated that extending the existing system will have an NPV of £57,000, although if the market expands significantly, this will be turned into a loss with an NPV of –£100,000 due to lost revenue. If the market does expand, replacing the system now has an NPV of £250,000 due to the benefits of being able to handle increased sales and other benefits such as improved management information. If sales do not increase, however, the benefits will be severely reduced and the project will suffer a loss with an NPV of –£50,000.

The company estimate the likelihood of the market increasing significantly at 20% – and, hence, the probability that it will not increase as 80%. This scenario can be represented as a tree structure as shown in Figure 3.5.

The analysis of a decision tree consists of evaluating the expected benefit of taking each path from a decision point (denoted by D in Figure 3.5). The expected value of each path is the sum of the value of each possible outcome multiplied by its probability of occurrence. The expected value of extending the system is therefore £40,000 ($75{,}000 \times 0.8 - 100{,}000 \times 0.2$) and the expected value of replacing the system £10,000 ($250{,}000 \times 0.2 - 50{,}000 \times 0.8$). IOE should therefore choose the option of extending the existing system.

This example illustrates the use of a decision tree to evaluate a simple decision at the start of a project. One of the great advantages of using decision trees to

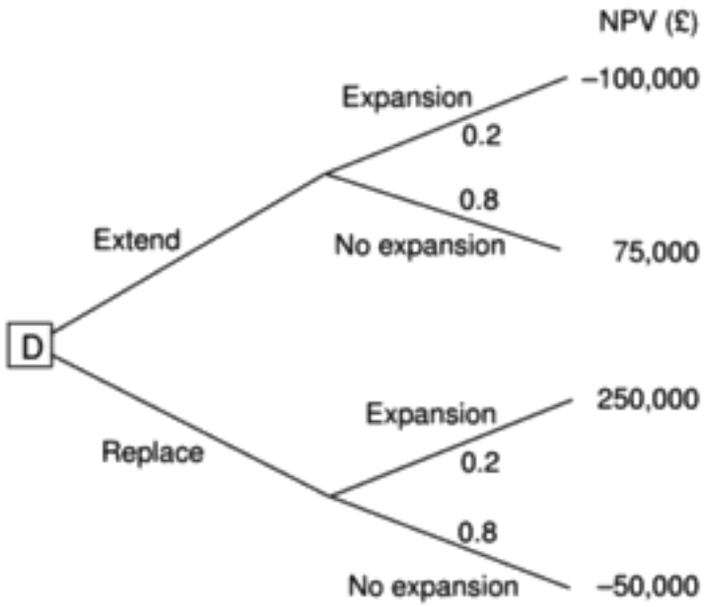


Figure 3.5 *A decision tree.*

model and analyse problems is the ease with which they can be extended. Figure 3.6 illustrates an extended version of Amanda's decision tree, which includes the possibility of a later decision should they decide to extend the system and then find there is an early market expansion.

The net present values shown in italic are those identified in Amanda's original decision tree shown in Figure 3.5.

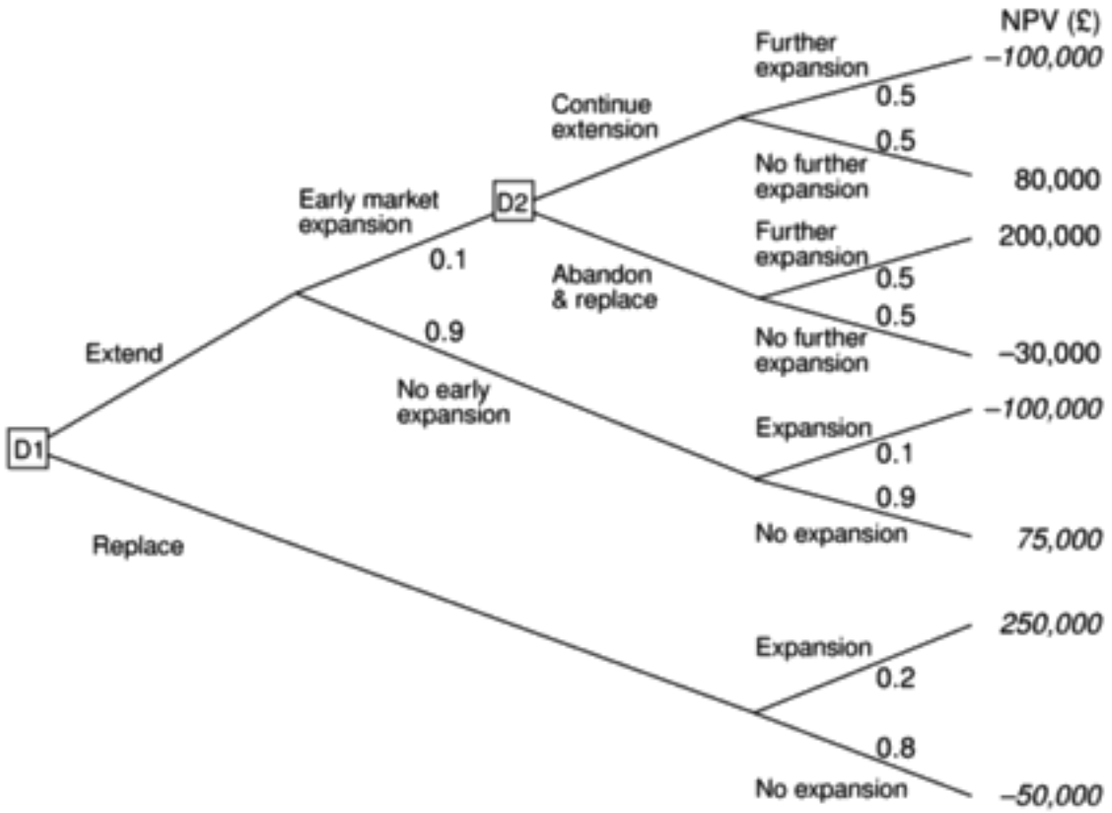


Figure 3.6 *An extension to Amanda's decision tree.*

3.8 Conclusion

Some of the key points in this chapter are:

- projects must be evaluated on strategic, technical and economic grounds;
- economic assessment involves the identification of all costs and income over the lifetime of the system, including its development and operation and checking that the total value of benefits exceeds total expenditure;
- money received in the future is worth less than the same amount of money in hand now, which may be invested to earn interest;
- the uncertainty surrounding estimates of future returns lowers their real value measured now;
- discounted cash flow techniques may be used to evaluate the present value of future cash flows taking account of interest rates and uncertainty;
- cost–benefit analysis techniques and decision trees provide tools for evaluating expected outcomes and choosing between alternative strategies.

3.9 Further exercises

1. Identify the major risks that could affect the success of the Brightmouth College Payroll project and try to rank them in order of importance.
2. Working in a group of three or four, imagine that you are about to embark upon a programming assignment as part of the assessed work for your course. Draw up a list of the risks that might affect the assignment outcome. Individually classify the importance and likelihood of each of those risk as high, medium or low. When you have done this compare your results and try to come up with an agreed project risk matrix.
3. Explain why discounted cash flow techniques provide better criteria for project selection than net profit or return on investment.
4. Consider the decision tree shown in Figure 3.6 and decide, given the additional possibilities, which option(s) IOE should choose.

Chapter 4

Selection of an appropriate project approach

OBJECTIVES

When you have completed this chapter you will be able to:

- take account of the characteristics of the system to be developed when planning a project;
- select an appropriate process model;
- make best use of the 'waterfall' process model where appropriate;
- reduce risks by the creation of appropriate prototypes;
- reduce other risks by implementing the project in increments.

4.1 Introduction

The development of software in-house suggests that the project has certain characteristics:

- the project team and the users belong to the same organization;
- the projects being considered slot into a portfolio of existing computer-based systems;
- the methodologies and technologies to be used are not selected by the project manager, but are dictated by local standards.

However, where a series of development projects is being carried out by a software house for different external customers, the methodologies and technologies to be used will have to be decided for each individual project. This decision-making process has been called 'technical planning' by some, although here we use the term 'project analysis'. Even where development is in-house, it is important to spend some time looking for any characteristics of the new project that might make us take a different approach from that used on previous projects. It is this analysis that is the subject of this chapter.

Martyn Ould in *Strategies for Software Engineering: the Management of Risk & Quality*, John Wiley & Sons, 1990, describes how technical planning was done at the software house, Praxis.

The relevant part of the Step Wise approach is Step 3: Analyse project characteristics. The selection of a particular process model will add new products to the Project Breakdown Structure or new activities to the activity network. This will create outputs for Step 4: Identify the products and activities of the project (see Figure 4.1).

In the remainder of this chapter we will firstly look at how the characteristics of a project will influence the approach to the planning of a project. We will then look at some of the most common *process models*, namely the waterfall approach, prototyping and incremental delivery.

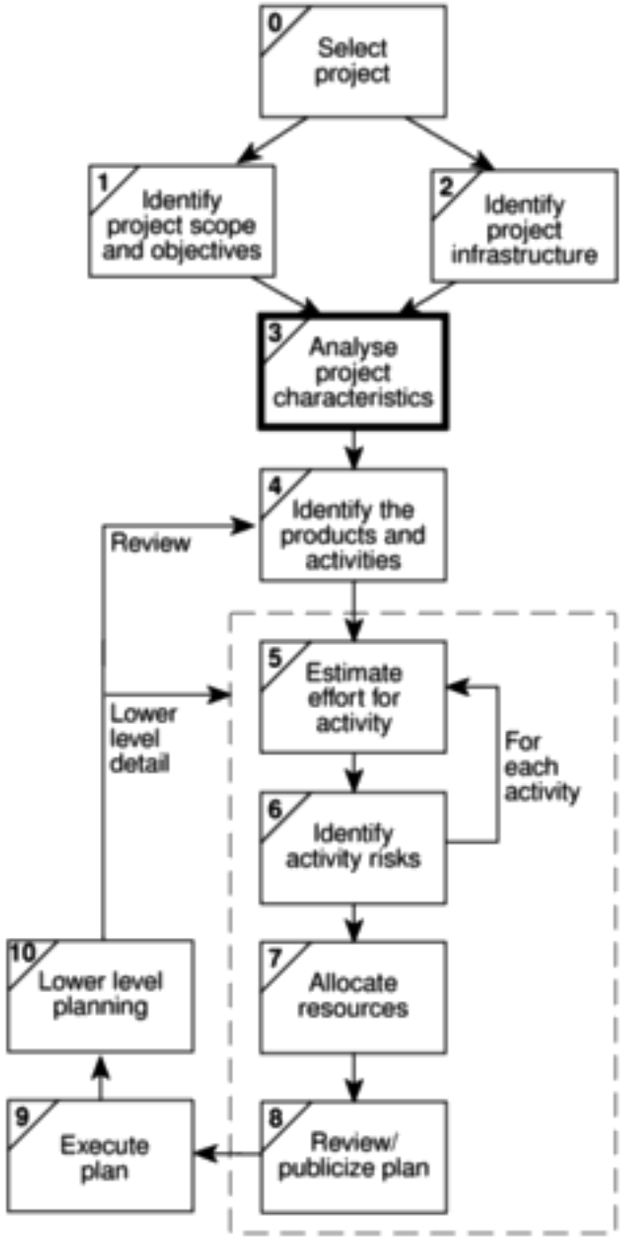


Figure 4.1 *Project analysis is the subject of Step 3.*

4.2 Choosing technologies

An outcome of project analysis will be the selection of the most appropriate methodologies and technologies. Methodologies include techniques like the various flavours of object-oriented (OO) development, SSADM and JSP (Jackson Structured Programming) while technologies might include an appropriate application-building environment, or the use of knowledge-based system tools.

As well as the products and activities, the chosen technology will influence the following aspects of a project:

- the training requirements for development staff;
- the types of staff to be recruited;
- the development environment – both hardware and software;
- system maintenance arrangements.

We are now going to describe some of the steps of project analysis.

Identify project as either objectives-driven or product-driven

You will recall from Chapter 1 that we distinguished between objectives-driven and product-driven projects. Very often a product-driven project will have been preceded by an objectives-driven project which chose the general software solution that is to be implemented.

There will be cases where things are so vague that even the objectives of the project are uncertain or are the subject of disagreement. People may be experiencing a lot of problems but no-one knows exactly what the solution to the problems might be. It could be that the IT specialists can provide help in some places but assistance from other specialisms is needed in others. In these kinds of situation a *soft systems* approach might need to be considered.

The soft systems approach is described in Checkland, P. and Scholes, J., *Soft systems methodology in action*, John Wiley and Sons, 1990.

Analyse other project characteristics

The sorts of question that would need to be asked include the following.

- **Is a data orientated or a control orientated system to be implemented?** 'Data orientated' systems generally mean information systems that will have a considerable database. 'Control orientated' systems refer to embedded control systems. These days it is not uncommon to have systems with components of both types.

We first introduced the difference between information systems and embedded systems in Chapter 1.

- **Will the software that is to be produced be a general package or application specific?** An example of a general package would be a spreadsheet or a word processing package. An application specific package could be, for example, an airline seat reservation system.
- **Is the system to be implemented of a particular type for which specific tools have been developed?** For example:

  - *does it involve concurrent processing?* – if so the use of techniques appropriate to the analysis and design of such systems would be considered;
  - *will the system to be created be knowledge-based?* – expert systems have a set of rules which result in some 'expert advice' when applied to a problem domain (sets of methods and tools have been developed to assist in the creation of such systems); or
  - will the system to be produced make heavy use of computer graphics?
- **Is the system to be created safety-critical?** For instance, could a malfunction in the system endanger human life?
- **What is the nature of the hardware/software environment in which the system will operate?** It might be that the environment in which the final software will operate is different from that in which it will be developed. Embedded software may be developed on a large development machine that has lots of supporting software tools in the way of compilers, debuggers, static analysers and so on, but might then be down-loaded to a small processor in the larger engineered product. A system destined for a personal computer will need a different approach to one destined for a main-frame or a client-server environment.

Exercise 4.1

How would you categorize each of the following systems according to the classification above?

(a) a payroll system
(b) a system to control a bottling plant
(c) a system that holds details of the plans of plant used by a water company to supply water to consumers
(d) a software application to support project managers
(e) a system used by lawyers to get hold of case law relating to company taxation.

Identify high level project risks

Chapter 3 has already touched on some aspects of risk, which are developed further in Chapter 8.

When we first embark on a project we might be expected to work out elaborate plans even though we are at least partially ignorant of many important factors that will affect the project. For example, until we do a detailed investigation of the users' requirements we will not be able to estimate how much effort will be needed to build a system to meet those requirements. The greater the uncertainties at the beginning of the project, the greater the risk that the project will be unsuccessful. Once we recognize a particular area of uncertainty we can, however, take steps to reduce its uncertainty.

One suggestion is that uncertainty can be associated with the *products*, *processes*, or *resources* associated with the project.

Product uncertainty Here we ask how well the requirements are understood. It might be that the users themselves are uncertain about what a proposed information system is to do. The government, say, might introduce a new form of taxation but the way this is going to operate in detail will not be known until a certain amount of case law has been built up. Some environments can change so quickly that what was a precise and valid statement of requirements rapidly becomes out of date.

Process uncertainty It might be that the project under consideration is the first where an organization has tried to use a method, such as SSADM or OMT, that is new to them. Perhaps a new application building tool is being used. Any change in the way that the systems are developed is going to introduce uncertainty.

OMT is an object-oriented design approach.

Resource uncertainty The main area of uncertainty here will almost surely be the availability of staff of the right ability and experience. A major influence on the degree of uncertainty in a project will be the sheer size of a project. The larger the number of resources needed or the longer the duration of the project, the more inherently risky it is likely to be.

Euromethod, which is reviewed briefly in Appendix C, distinguishes between factors that increase *uncertainty*, for example, continually changing requirements, and those that increase *complexity*, for example, software size. This is because it suggests different strategies to deal with the two distinct types of risk.

Exercise 4.2

At IOE, Amanda has identified possible staff resistance as a risk to the maintenance group accounts project. Would you classify this as a product, process or resource risk? Perhaps it does not fit into any of these categories and some other is needed.

Brigette at Brightmouth College identified the possibility that no suitable payroll package would be available on the market as a risk. What other risks might be inherent in the Brightmouth College payroll project?

Take into account user requirements concerning implementation

A user organization lays down standards that have to be adopted by any contractor providing software for them. For example the UK Civil Service favours the SSADM standard where information systems are being developed.

It is common for organizations to specify that suppliers of software have BS EN ISO 9001:1994 or TickIT accreditation. This will affect the way projects are conducted.

Chapter 12, Software quality, discusses BS EN ISO 9001 and TickIT.

Select general life cycle approach

Control systems A real-time system will have to be implemented using an appropriate methodology, for example, Mascot. Real-time systems that employ concurrent processing will use techniques such as Petri nets.

Information systems Similarly, an information system will need a methodology, such as SSADM or Information Engineering, that matches that type of environment. SSADM will be especially appropriate where the project will employ a large number of development staff whose work will need to be co-ordinated: the method lays down in detail what needs to be done and what products need to be created at each step. Team members would therefore know exactly what is expected of them.

General applications Where the software to be produced is for the general market rather than for a specific application and user, then a methodology such as SSADM would have to be thought about very carefully. This is because the framers of the method make the assumption that a specific user exists. Some parts in the method also assume that there is an existing system that can be analysed to yield the logical features of the new, computer-based, system.

Specialized techniques These have been invented to expedite the development of, for example, *knowledge-based systems* where there are a number of specialized tools and logic based programming languages that can be used to implement this type of system. Similarly a number of specialized techniques and tools have been developed to assist in the development of *graphics-based systems*.

Hardware environment The environment in which the system is to operate can put constraints on the way it is to be implemented. For instance, the need for a fast response time or for the software to take up only a small part of computer memory may mean that only low-level programming languages can be used – particularly in real-time and embedded systems.

Safety-critical systems Where safety and reliability are of the essence, it might be possible to justify the additional expense of a formal specification using a notation such as Z or VDM. Really critical systems call for expensive measures such as having independent teams develop parallel systems with the same functionality. The parallel systems can then run concurrently when the application is in operation so that the results of each of the parallel systems can be cross-checked.

The implications of prototyping and the incremental approach are explored later in the chapter.

Imprecise requirements Uncertainties or a *novel hardware/software platform* may mean that a *prototyping* approach should be considered. If the environment in which the system is to be implemented is a rapidly changing one, then serious consideration would need to be given to *incremental delivery*. If the users have *uncertain objectives* in connection with the project, then a *soft systems* approach might be desirable.

Exercise 4.3

Bearing in mind the discussion above, what, in broad outline, is the most suitable approach for each of the following?

(a) a system that calculates the amount of a drug that should be administered to a patient who has a particular complaint;

(b) a system to administer a student loans scheme;
(c) a system to control trains on a railway network.

4.3 Technical plan contents list

The analysis described above will produce a number of practical requirements that will be fed into the next stage of the planning process. These requirements might add activities to the project and might involve the acquisition of items of software or hardware or the adoption of particular methodologies which might require staff training. As these recommendations imply certain costs, they should be recorded formally.

A preliminary version of this technical plan can be produced by a software house to help in the preparation of the bid for a contract. In some cases, it may actually be shown to the potential customer in order to show the basis for the bid price and generally to impress the customer with the soundness of the approach the software house intends to adopt.

The technical plan is likely to have the following contents.

1. Introduction and summary of constraints:
   (a) character of the system to be developed;
   (b) risks and uncertainties of the project;
   (c) user requirements concerning implementation.
2. Recommended approach:
   (a) selected methodology or process model;
   (b) development methods;
   (c) required software tools;
   (d) target hardware/software environment.
3. Implementation:
   (a) required development environment;
   (b) required maintenance environment;
   (c) required training.
4. Implications:
   (a) project products and activities – these will have an effect on the schedule duration and overall project effort;
   (b) financial – this report will be used to produce costings.

4.4 Choice of process models

The word ‘process’ is sometimes used to emphasize the idea of a system *in action*. In order to achieve an outcome, the system will have to execute one or more activities: this is its process. This idea can be applied to the development of computer-based systems where a number of interrelated activities have to be

undertaken to create a final product. These activities can be organized in different ways and we can call these *process models*.

A major part of the planning will be the choosing of the development methods to be used and the slotting of these into an overall process model.

The planner needs not only to select methods but also to specify how the method is to be applied. With methods such as SSADM, there is a considerable degree of choice about how it is to be applied: not all parts of SSADM are compulsory. Many student projects have the rather basic failing that at the planning stage they claim that, say, SSADM is to be used: in the event, all that is produced are a few SSADM fragments such as a top level data flow diagram and a preliminary logical data structure diagram. If this is all the particular project requires, it should be stated at the outset.

4.5 Structured methods

Although some 'object-oriented' specialists may object(!), we include the OO approach as a structured method – after all, we hope it is not unstructured. Structured methods are made up of sets of steps and rules, which, when applied produce system products such as data flow diagrams. Each of these products is carefully documented. Such methods are often time consuming compared to more intuitive approaches and this implies some additional cost. The pay-off is such things as a less error prone and more maintainable final system. This balance of costs and benefits is more likely to be justified on a large project involving many developers and users. This is not to say that smaller projects cannot justify the use of such methods.

4.6 Rapid application development

It might be thought that users would generally welcome the more professional approach that structured methods imply. However, customers of IS/IT are concerned with getting working business applications delivered quickly and at less cost and often see structured methods as unnecessarily bureaucratic and slow. A response to this has been *rapid application development* (RAD).

The RAD approach does not preclude the use of some elements of structured methods such as Logical Data Structure diagrams but also adopts tactics such as *joint application development* (JAD) workshops. In these workshops, developers and users work together intensively for, say, three to five days and identify and agree fully documented business requirements. Often, these workshops are conducted away from the normal business and development environments in *clean rooms*, that is, special conference rooms free from outside interruption and suitably furnished with white boards and other aids to communication. This is based on the belief that communication and negotiation that might take several weeks or months via a conventional approach of formal reports and proposals and counter-proposals can be speeded up in these hothouse conditions.

Another practice associated with RAD is *time-boxing* where the scope of a project is rigidly constrained by a pre-agreed and relatively close absolute deadline. Requirements that have to be omitted in order to meet this deadline are considered in later time-boxes.

Some of these ideas will be considered further in our later discussions on prototyping and the incremental approach.

4.7 The waterfall model

This is the 'classical' model of system development. An alternative name for this model is the *one-shot* approach. As can be seen from the example in Figure 4.2, there is a sequence of activities working from top to bottom. The diagram shows some arrows pointing upwards and backwards. This indicates that a later stage might reveal the need for some extra work at an earlier stage, but this should definitely be the exception rather the rule. After all, the flow of a waterfall should be downwards with the possibility of just a little splashing back. The limited scope for iteration is in fact one of the strengths of this process model. With a large project you want to avoid having to go back and rework tasks that you thought had been completed. For a start, having to reopen what was previously thought to be a completed activity plays havoc with promised completion dates.

The first description of this approach is said to be that of H. D. Bennington in 'Production of Large Computer Programs' in 1956. This was reprinted in 1983 in *Annals of the History of Computing* 5(4).

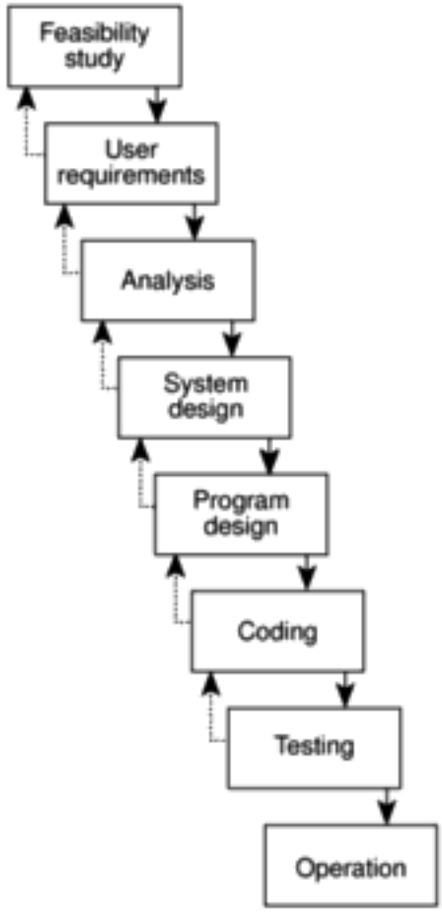


Figure 4.2 *The waterfall model.*

We contend that there is nothing intrinsically wrong with the waterfall approach, even though more recent writers have suggested different models. It is the ideal that the project manager strives for. The waterfall approach allows project completion times to be forecast with more confidence than is the case with some more iterative approaches and this allows projects to be controlled effectively. However, where there is uncertainty about how a system is to be implemented, and unfortunately there very often is, a more flexible, iterative, approach may be required.

4.8 The V-process model

Figure 4.3 gives a diagrammatic representation of this model. This is an elaboration of the waterfall model and stresses the necessity for validation activities that match the activities that create the products of the project.

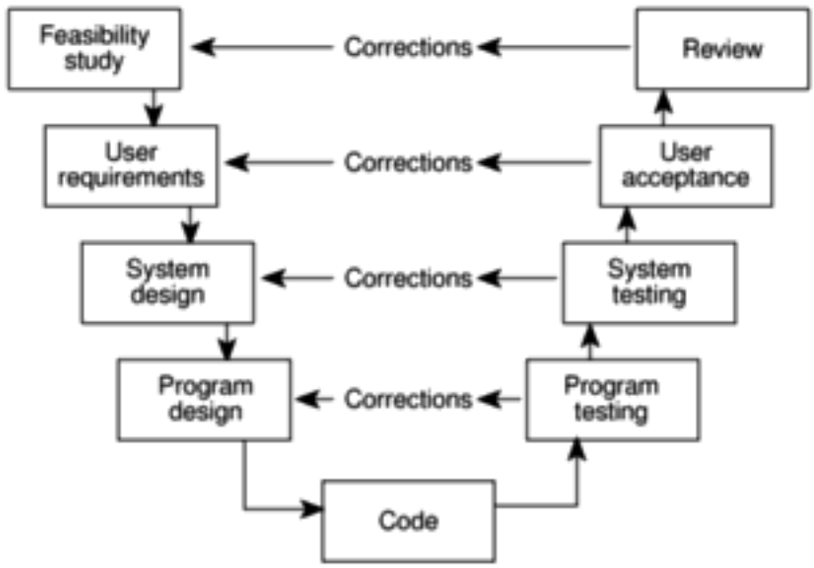


Figure 4.3 *The V-process model.*

The V-process model can be seen as expanding the testing activity in the waterfall model. Each step has a matching validation process that can, where defects are found, cause a loop back to the corresponding development stage and a reworking of the succeeding steps. Ideally, this feeding back should only occur where a discrepancy has been found between what was specified by a particular activity and what was actually implemented in the next lower activity on the descent of the V loop. For example, the system designer might have written that a calculation be carried out in a certain way. The person who structured the software that fulfilled this design might have misunderstood what was required. At system testing stage, the system designer would carry out checks that ensure that the software is doing what was specified in the design document and would discover the program designer's misreading of that document. Only corrections should be

fed back, not the system designer's second thoughts, otherwise the project would be slipping into an 'evolutionary prototyping' approach.

Exercise 4.4

Figure 4.3 shows the V-process model. The review that is held after the system has been implemented is shown as possibly feeding corrections back to the feasibility study which was probably conducted months or years before. How would this work in practice?

4.9 The spiral model

It could be argued that this is another way of looking at the basic waterfall model. In the waterfall model, there is a possible escape at the end of any of the activities in the sequence. A feasibility study might decide that the implementation of a proposed system would be beneficial. The management therefore authorize work on the detailed collection and analysis of user requirements. Some analysis, for instance the interviewing of users, might already have taken place at the feasibility stage, but a more thorough investigation is now launched. This might reveal that in fact the costs of implementing the system would be higher than originally estimated and lead managers to decide to abandon the project.

The original ideas behind the spiral model can be found in B. W. Boehm's 1988 paper 'A spiral model of software development and enhancement' in *IEEE Computer* 21(5).

A greater level of detail is considered at each stage of the project and a greater degree of confidence about the probability of success for the project should be justified. This can be portrayed as a loop or a spiral where the system to be implemented is considered in more and more detail in each sweep and an evaluation process is undertaken before the next iteration is embarked upon. Figure 4.4 illustrates how SSADM can be interpreted in such a way.

4.10 Software prototyping

A prototype is a working model of one or more aspects of the projected system. It is constructed and tested quickly and inexpensively in order to test out assumptions.

Prototypes can be classified as throw-away, evolutionary or incremental.

Throw-away prototypes Here the prototype is used only to test out some ideas and is then discarded when the development of the operational system is commenced. The prototype could be developed using a different software environment (for example, a 4GL as opposed to a 3GL for the final system where machine efficiency is important) or even on a different kind of hardware platform.

Evolutionary prototypes The prototype is developed and modified until it is finally in a state where it can become the operational system. In this case, the standards that are used to develop the software have to be carefully considered.

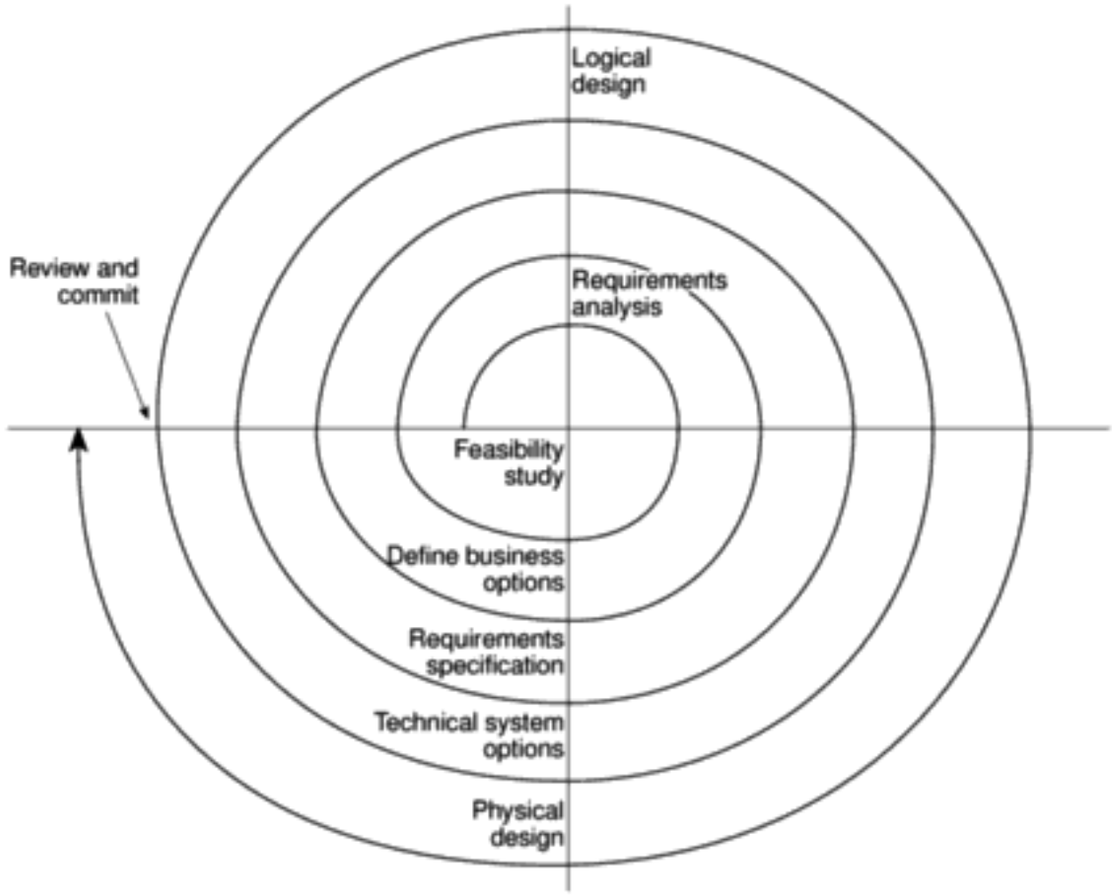


Figure 4.4 *The application of the spiral model to SSADM version 4.*

Incremental prototypes It could be argued that this is, strictly speaking, not prototyping. The operational system is developed and implemented in small stages so that the feed-back from the earlier stages can influence the development of the later stages.

A good book on the general topic of prototyping is R. Vonk's *Prototyping: the effective use of CASE Technology*, Prentice Hall, 1990.

Some of the reasons that have been put forward for prototyping are the following.

Learning by doing When we have just done something for the first time we can usually look back and see where we have made mistakes.

Improved communication Users are often reluctant to read the massive documents produced by structured methods. Even if they do read this documentation, they do not get a feel for how the system is likely to work in practice.

The most important justification for a prototype is the need to reduce uncertainty by conducting an experiment.

Improved user involvement The users may be more actively involved in design decisions about the new system.

Clarification of partially-known requirements Where there is no existing system to mimic, users can often get a better idea of what might be useful to them in a potential system by trying out prototypes.

Demonstration of the consistency and completeness of a specification Any mechanism that attempts to implement a specification on a computer is likely to uncover ambiguities and omissions.

Reduced need for documentation Because a working prototype can be examined, there is less need for detailed documentation. Some may argue, however, that this is a very dangerous suggestion.

Reduced maintenance costs (that is, changes after the system goes live) If the user is unable to suggest modifications at the prototyping stage the chances are that they will ask for the changes as modifications to the operational system. This reduction of maintenance costs is the main plank in the financial case for creating prototypes.

Feature constraint If an application building tool is used, then the prototype will tend to have features that are easily implemented by that tool. A paper-based design might suggest features that are expensive to implement.

Production of expected results The problem with creating test runs is generally not the creation of the test data but the accurate calculation of the expected results. A prototype can be of assistance here.

Software prototyping is not without its drawbacks and dangers, however.

Users sometimes misunderstand the role of the prototype For example, they might expect the prototype to be as robust as an operational system when incorrect data is entered or they might expect the prototype to have as fast a response as the operational system, although this was not the intention.

Lack of project standards possible Evolutionary prototyping could just be an excuse for a sloppy 'hack it out and see what happens' approach.

Lack of control It is sometimes difficult to control the prototyping cycle as the driving force could be the users' propensity to try out new things.

Additional expense Building and exercising a prototype will incur additional expenses. The additional expense might be less than expected, however, because many analysis and design tasks would have to be undertaken anyway. Some research suggests that typically there is a 10% extra cost to produce a prototype.

Machine efficiency A system built through prototyping, while sensitive to the users' needs, might not be as efficient in machine terms as one developed using more conventional methods.

Close proximity of developers Prototyping often means that code developers have to be sited close to the users. One trend has been for organizations in developed countries to get program coding done cheaply in Third World countries such as India. Prototyping generally will not allow this.

4.11 Other ways of categorizing prototypes

What is being learnt?

The most important reason for having a prototype is that there is a need to learn about an area of uncertainty. For any prototype it is essential that the project managers define at the outset what it is intended to learn from the prototype.

This has a particular relevance to student projects. Students often realize that the software that they are to write as part of their project could not safely be used by real users. They therefore call the software a 'prototype'. However, if it is a real prototype then they must:

- specify what they hope to learn from the prototype;
- plan how the prototype is to be evaluated;
- report on what has actually been learnt.

Prototypes may be used to find out how a new development technique can be used. This would be the case where a new methodology is being used in a pilot scheme. Alternatively, the development methods might be well-known, but the nature of the application might be uncertain.

Different projects will have uncertainties at different stages. Prototypes can therefore be used at different stages. A prototype might be used, for instance, at the requirements gathering stage to pin down requirements that seem blurred and shifting. A prototype might, on the other hand, be used at the design stage to test out the users' ability to navigate through a sequence of input screens.

To what extent is the prototyping to be done?

It would be unusual for the whole of the application to be prototyped. The prototyping might take one of the following forms, which simulates only some aspects of the target application.

Mock-ups For example, copies of the screens that the system is to use are shown to the users on a terminal, but the screens cannot actually be used.

Simulated interaction For example, the user can type in a request to access a record and the system will respond by showing the details of a record, but the details shown are always the same and no access is made to a database.

Partial working model:

- *vertical* – some features are prototyped fully
- *horizontal* – all features are prototyped but not in detail (for example, there might not be a full validation of input).

What is being prototyped?

The human–computer interface With information systems, what the system is to do has usually been established and agreed by management at a fairly early

stage in the development of the system. Prototyping tends to be confined to establishing the precise way in which the operators are to interact with the system. In this case, it is important that the physical vehicle for the prototype be as similar as possible to the operational system.

The functionality of the system In other cases, the precise way that the system should function internally will not be known. This will particularly be the case where a computer model of some real-world phenomenon is being developed. The algorithms used need to be repeatedly adjusted until they satisfactorily imitate the behaviour they should be modelling.

Exercise 4.5

At what stage of a system development project (for example, feasibility study, requirements analysis etc.) would a prototype be useful as a means of reducing the following uncertainties?

(a) There is a proposal that the senior managers of an insurance company have personal access to management information through an executive information system installed on personal computers located on their desks. Such a system would be costly to set up and there is some doubt about whether the managers would actually use the system.

(b) A computer system is to support sales office staff who take phone calls from members of the public enquiring about motor insurance and give quotations over the phone.

(c) The insurance company is considering implementing the telephone sales system using the system development features supplied by Microsoft Access. They are not sure, at the moment, that it can provide the kind of interface that would be needed and are also concerned about the possible response times of a system developed using Microsoft Access.

4.12 Tools

Special tools are not essential for prototyping but some have made it more practicable.

Application building tools have many features that allow simple computer-based information systems to be set up quickly so that they can be demonstrated to the staff who will use them. An application written in a 3GL becomes more and more complicated as changes accumulate in its code and this makes the software more difficult to alter safely, while applications implemented using application builders that are often form- and table-driven remain flexible.

It has been suggested that the ease of use of some 4GL application builders and the increasing IT awareness of end users might allow end users to create their own prototypes.

Chapter 9 explores configuration management further.

Where a prototype is being constantly tested by the users and modified by the developers there is a need to be able to control the different versions being produced and where necessary to back-track to previous versions.

4.13 A prototyping example

Details of the study can be found in Mayhew, P. J., Worseley, C. J. & Dearnley, P. A. 'Control of Software Prototyping Process: Change Classification Approach'. *Information and Systems Technology* 13(2) 59-66 published in 1989.

The use of prototyping on the COMET Commercial Estimating System project at the Royal Dockyards was the subject of a study, information about which has been published. The Royal Dockyards had been transferred to a system of 'commercial management' where they had to produce estimates of the cost of doing tasks that could then be compared against the cost of contracting the work out. Staff at the dockyards were inexperienced with this method of working.

It was decided to implement a computer system to support the estimating process and it was realized that the human–computer interface would be important. For this reason, the software house implementing the system suggested a prototyping approach

Among the questions that had to be decided was who should be present at evaluation sessions. If managers were there, would they dominate the proceedings at the expense of those who would actually use the system? Furthermore, if the designers were present they might well feel defensive about the system presented and try and argue against changes suggested.

The results of the prototype

The impact of prototyping may be judged by the fact that although the preliminary design had been done using SSADM, as a result of the evaluation of the prototype the number of screens in the system was doubled.

A major problem was that of controlling changes to the prototype. In order to record and control the changes suggested by users, the changes were categorized into three types.

Cosmetic (about 35% of changes). These were simply changes to the layout of the screen. They were:

- implemented;
- recorded.

Local (about 60% of changes). These involved changes to the way that the screen was processed but did not affect other parts of the system. They were:

- implemented;
- recorded;
- backed-up so that they could removed at a later stage if necessary;
- inspected retrospectively.

Inspections are discussed in Chapter 12.

Global (about 5% of changes), These were changes that affected more than one part of the processing. All changes here had to be the subject of a design review before they could be implemented.

4.14 Incremental delivery

This is similar to the 'incremental prototyping' approach mentioned above. One of the most prominent advocates of this approach is Tom Gilb. The approach involves breaking the system down into small components which are then implemented and delivered in sequence. Each component that is delivered must actually give some benefit to the user. Figure 4.5 gives a general idea of the approach.

Principles of Software Engineering Management by Tom Gilb, published by Addison-Wesley in 1988, argues strongly in favour of this approach.

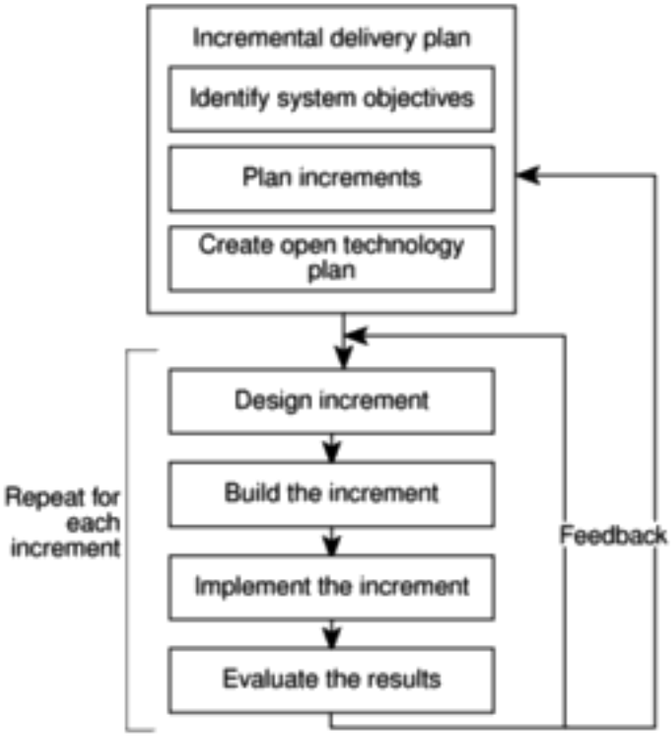


Figure 4.5 *Intentional incremental delivery.*

Advantages of this approach

These are some of the justifications given for the approach:

- the feedback from early increments can influence the later stages;
- the possibility of changes in requirements is not so great as with large monolithic projects because of the shorter timespan between the design of a component and its delivery;
- users get benefits earlier than with a conventional approach;
- early delivery of some useful components improves cash flow, because you get some return on investment early on;

- smaller sub-projects are easier to control and manage;
- 'gold-plating', the requesting of features that are unnecessary and not in fact used, should be less as users will know that they get more than one opportunity to make their requirements known: if a feature is not in the current increment then it can be included in the next;
- the project can be temporarily abandoned if more urgent work crops up;
- job satisfaction is increased for developers who see their labours bearing fruit at regular, short, intervals.

Disadvantages

On the other hand these disadvantages have been put forward:

- 'software breakage', that is, later increments might require the earlier increments to be modified;
- developers might be more productive working on one large system than on a series of smaller ones.

The incremental delivery plan

The process of planning the increments of a project as described by Gilb has similarities with strategic planning described in the previous chapter.

The content of each increment and the order in which the increments are to be delivered to the users of the system have to be planned at the outset.

Basically the same process has to be undertaken as in strategic planning but at a more detailed level where the attention is given to increments of a user application rather than whole applications. The elements of the incremental plan are the *system objectives*, *incremental plan* and the *open technology plan*.

Identify system objectives

The purpose is to give an idea of the 'big picture', that is, the overall objectives that the system is to achieve. These can then expanded into more specific functional goals and quality goals.

Functional goals will include:

- objectives it is intended to achieve;
- jobs the system is to do;
- computer/non-computer functions to achieve them.

Chapter 12 discusses software quality characteristics.

In addition, measurable quality characteristics should be defined, such as reliability, response and security. This reflects Tom Gilb's concern that system developers always keep sight of the objectives that they are trying to achieve on behalf of their clients. In the quickly changing environment of an application, individual requirements may change over the course of the project, but the objectives should not.

Plan increments

Having defined the overall objectives, the next stage is to plan the increments using the following guidelines:

- steps typically should consist of 1% to 5% of the total project;
- non-computer steps may be included – but these must deliver benefits directly to the users;
- ideally, an increment should take one month or less and should never take more than three months;
- each increment should deliver some benefit to the user;
- some increments will be physically dependent on others;
- value-to-cost ratios may be used to decide priorities (see below).

Very often a new system will be replacing an old computer system and the first increments may use parts of the old system. For example, the data for the database of the new system may initially be obtained from the old system's standing files.

Which steps should be first? Some steps might be prerequisites because of physical dependencies but others can be in any order. Value-to-cost ratios can be used to establish the order in which increments are to be developed. The customer is asked to rate each increment with a score in the range 1–10 in terms of its value. The developers also rate the cost of developing each of the increments with a score in the range 0–10. This might seem a rather crude way of evaluating costs and benefits, but people are often unwilling to be more precise. By then dividing the value rating by the cost rating, a rating which indicates the relative 'value for money' of each increment is derived.

The value to cost ratio = V/C where V is a score 1–10 representing value to customer and C is a score 0–10 representing cost.

Table 4.1 *Ranking by value to cost ratio*

Step	*Value*	*Cost*	*Ratio*	*Rank*
Profit reports	9	1	9	(2nd)
Online database	1	9	0.11	(6th)
Ad hoc enquiry	5	5	1	(4th)
Production sequence plans	2	8	0.25	(5th)
Purchasing profit factors	9	4	2.25	(3rd)
Clerical procedures	0	7	0	(7th)
Profit based pay for managers	9	0	∞	(1st)

Create open technology plan

If the system is to be able to cope with new components being continually added then it has to be built so that it is extendible, portable and maintainable.

As a minimum this will require the use of:

- a standard high level language;
- a standard operating system;
- small modules;
- variable parameters, for example, items such as organization name, department names and charge rates are held in a parameter file that can be amended without programmer intervention;
- a standard database management system.

These are all things that might be expected as a matter of course in a modern IS development environment.

4.15 An incremental example

Tom Gilb describes a project where a software house negotiated a fixed price contract with a three-month delivery time with the Swedish Government to supply a system to support map-making. It later became apparent that the original estimate of effort upon which the bid was based was probably about half what the real effort would be.

The project was replanned so that it was divided into ten increments, each supplying something of use to the customer. The final increments were not available until three months after the contract's delivery date. The customer was not in fact unhappy about this as the most important parts of the system had actually been delivered early.

4.16 Selecting the most appropriate process model

See Appendixes C and D for overviews of Euromethod and ISO 12207, both of which offer advice on process model selection.

Both Euromethod and ISO 12207 provide advice on this.

Euromethod distinguishes between the *construction* of an application and its *installation*. It is possible to use different approaches for these two stages. For example, an application could be constructed using a one-shot strategy but then be released to its users in increments. The only combinations of construction and installation strategies that are not feasible are evolutionary installation with any other construction approach than evolutionary.

In general, Euromethod suggests that where uncertainty is high then an evolutionary approach is to be favoured. An example of uncertainty would be where the users' requirements are not clearly defined. Where the requirements are relatively certain but there are many complexities, for example, where there is a large embedded system that is going to need a large amount of code, then an incremental approach might be favoured. Where deadlines are tight, then either an evolutionary or an incremental approach is favoured over a one-shot strategy, as both tactics should allow at least something to be delivered at the deadline, even

if it is not all that was originally promised. Students about to plan final year projects would do well to note this.

4.17 Conclusion

This chapter has stressed the need to examine each project carefully to see if it has characteristics that suggest a particular approach or process model. These characteristics will also suggest the addition of specific activities to the project plan.

The classic waterfall process model, which attempts to minimize iteration, should lead to projects that are easy to control. Unfortunately many projects do not lend themselves to this structure. Prototyping often reduces project uncertainties by allowing knowledge to be bought through experimentation. The incremental approach encourages the execution of a series of small, manageable, 'mini-projects' but does have some costs.

4.18 Further exercises

1. A bank which provides loans to home buyers has a long history of implementing computer-based information systems to support the work of its branches. It uses a proprietary structured systems analysis and design method. It has been decided to create a computer model of the property market. This would attempt for example to calculate the effect of changes of interest rates on house values. There is some concern that the usual methodology used for IS development would not be appropriate for the new project.

   (a) Why might there be this concern and what other approaches should be considered?
   (b) Outline a plan for the development of the system that illustrates the application of your preferred methodology for this project.

2. A software application is to be designed and built to assist in software cost estimation. It responds to certain input parameters and produces initial cost estimates to be used at bidding time.

   (a) It has been suggested that a software prototype would be of value in these circumstances. Explain why this might be.
   (b) Discuss how such prototyping could be controlled to ensure that it is conducted in an orderly and effective way and within a specified time span.

3. An invoicing system is to have the following components: amend invoice, produce invoice, produce monthly statements, record cash payment, clear paid invoices from database, create customer records, delete customer.

   (a) What physical dependencies govern the order in which these transactions are implemented?

(b) How could the system be broken down into increments which would be of some value to the users. Hint – think about the problems of taking existing details onto a database when a system is first implemented.

4. What are the features of the following that contribute to an open systems architecture as recommended by Tom Gilb:

   (a) the UNIX™ operating system;
   (b) SQL;
   (c) C++;
   (d) Jackson Structured Programming.