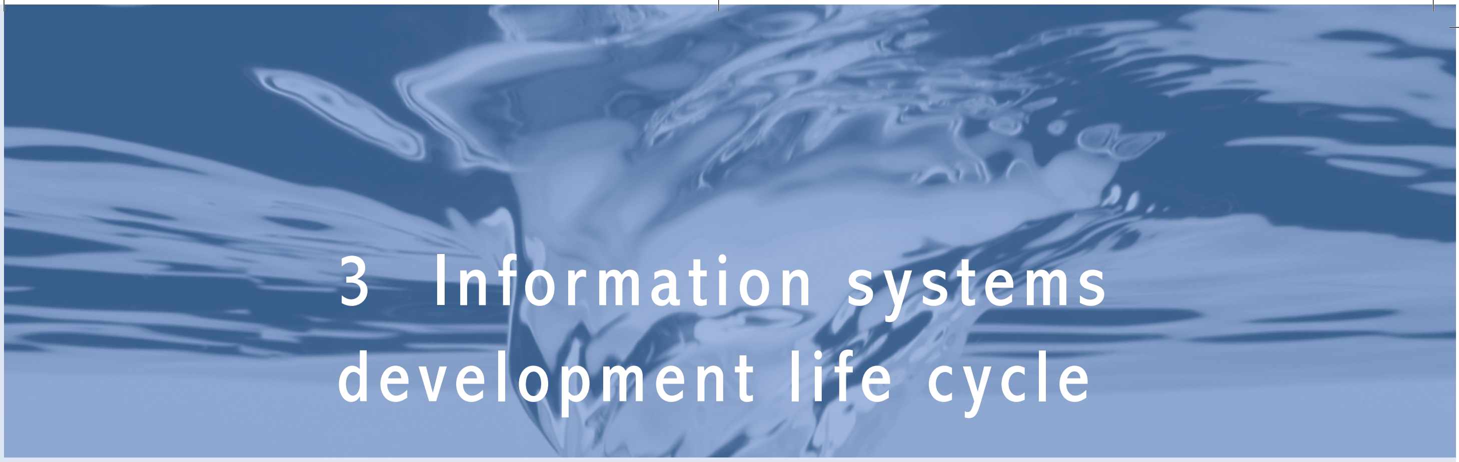
Part II: The life cycle approach

Contents

MG12454 ch03.qxp 2/2/06 14:56 Page 30

In Part II we introduce the information systems development life cycle, which is a general approach to developing systems. It has been the basis of many systems development projects since the 1970s. The phases in the approach are described along with the strengths of the approach. We use this approach as an exemplar and introduce methodologies, techniques, and tools by means of this approach. However, we also discuss its weaknesses, or at least problems relating to the way it has been used in practice, as they lead on to the alternative approaches to developing information systems, which are introduced in Part III of the book. Many of the alternative themes discussed there have in their roots attempts to address some of these weaknesses.



3 Information systems development life cycle

3.1 Information systems development life cycle (SDLC)

The SDLC has had an enormous influence as a general approach to develop information systems. Although there are many variants, it has the following basic structure:

- feasibility study;
- system investigation;
- systems analysis;
- systems design;
- implementation;
- review and maintenance.

These stages together are frequently referred to as 'conventional systems analysis', 'traditional systems analysis', 'information systems development life cycle', or, more frequently in the USA, the **waterfall model**. The term 'life cycle' indicates the staged nature of the process. Further, by the time the review stage comes, the information system may be found to be inadequate and it may not be long before the process starts again with a feasibility study to develop another information system to replace it. In the next few sections we describe each stage in turn.

1 Feasibility study

Among other reasons, a computer system might be contemplated to replace an old system because increasing workloads have overloaded the present system, suitable staff are expensive and difficult to recruit, advancing technology leads to new possibilities, there is a change in the type of work, or there are frequent errors. The next stage is to look in more detail at the present system and then to determine the requirements of the new one.

The feasibility study looks at the present system, the requirements that it was intended to meet, problems in meeting these requirements, new requirements that have come to light since it was first implemented, and briefly investigates alternative solutions. These must be within the terms of reference given to the analyst relating to the boundaries of the system and constraints, particularly those associated with resources. The alternatives suggested might include leaving things alone and improved manual as well as computer solutions.

For each of these, a description is given in terms of the technical, human, organizational, and economic costs and benefits of developing and operating the system. So, any proposed system must be feasible:

- *legally*, that is, it does not infringe any national or, if relevant, international company law;
- *organizationally and socially*, that is, it is acceptable for the organization and its staff, particularly if it involves major changes to the way in which the organization presently carries out its processing;
- *technically*, that is, it can be supported by the technology available and there is sufficient expertise to build it;
- *economically*, that is, it is financially affordable and the expense justifiable.

Of the possible alternatives, a 'recommended solution' is proposed with an outline functional specification. This information is given to management as a formal report and often through an oral presentation by the systems analysts to management who will then decide whether to accept the recommendations of the analysts. This is one of the decision points in the SDLC – whether to proceed or not.

2 Systems investigation

If management has given its approval to proceed, the next stage is a detailed fact-finding phase. This purports to be a thorough investigation of the application area. The information obtained will be much more detailed than that recorded in the feasibility report. It will look at:

- the functional requirements of the existing system (if there is one) and whether these requirements are being achieved;
- the requirements of the new system as there may be new situations or opportunities that suggest altered requirements;
- any constraints imposed;
- the range of data types and volumes that have to be processed;
- exception conditions;
- problems of the present working methods.

These facts are gained by interviewing personnel (both management and operational staff), through the use of questionnaires, by direct observation of the application area of interest, by sampling, and by looking at records and other written material related to the application area:

- *observation* can give a useful insight into the problems, work conditions, bottlenecks, and methods of work;
- *interviewing*, which may be conducted with individuals and groups of users, is usually the most helpful technique for establishing and verifying information, and provides an opportunity to meet the users and to start to overcome possible resistance to change;

- *questionnaires* are usually used where similar types of information need to be obtained from a large number of respondents or from remote locations;
- *searching records and documentation* may highlight problems, but the analyst has to be aware that the documentation may be out of date;
- *sampling* may be used but often requires specialist help from people with statistical and other skills.

A great deal of skill is required to use any of these effectively, and results need to be cross-checked by using a number of these approaches. It may be possible to find out about similar systems implemented elsewhere as the experience of others could be invaluable.

3 Systems analysis

Armed with the facts, the systems analyst proceeds to the systems analysis phase and analyses the present system by asking such questions as:

- Why do the problems exist?
- Why were certain methods of work adopted?
- Are there alternative methods?
- What are the likely growth rates of data?

In other words, it is an attempt to understand all aspects of the present system and why it developed as it did, and eventually indicate how things might be improved by any new system. So, systems analysis provides pointers to the new design. In particular, the analysts will emphasize the need to ascertain what are the requirements for the new system (see Section 5.6).

4 Systems design

Although usually modelled on the design suggested at the feasibility study stage, the new facts may lead to the analyst adopting a rather different design to that proposed at that time. Much will depend on the willingness to be thorough in the investigation phase and questioning in the analysis phase. Typically, however, the new design might be similar to the previous system, but avoiding the problems that occurred with the old system and without including any new ones. Sometimes, the new system is somewhat more radical.

This stage involves the design of both the computer and manual parts of the system. The design documentation set will contain details of:

- input data and how the data is to be captured (entered in the system);
- outputs of the system;
- processes, many carried out by computer programs, involved in converting the inputs to the outputs;
- structure of the computer and manual files that might be referenced in the system;
- security and backup provisions to be made;
- systems testing and implementation plans.

5 Implementation

Following the systems design phase are the various procedures that lead to the implementation of the new system. If the design includes computer programs, these have to be written and tested. New hardware and software systems need to be purchased and installed if they are not available in the organization at present. It is important that all aspects of the system are proven before **cutover** to the new system, otherwise failure will cause a lack of confidence in this and, possibly, future computer applications. The design and coding of the programs might be carried out by computer programmers. Alternatively, application packages might be purchased to form part of the final system. In this approach, the analysis and programming functions are separate tasks carried out by different people.

A major aspect of this phase is that of **quality control**. The manual procedures, along with the hardware and software, need to be tested to the satisfaction of users as well as analysts. The users need to be comfortable with the new methods. The departmental staff can practise using the system and any difficulties experienced should be ironed out. The **education and training** of user staff is therefore an important element of this phase. Without thorough training, users will be unfamiliar with the new system and unlikely to cope with the new approach (unless it is very similar to the old system).

Documentation, such as the operations and user manuals, will be produced and approved and the live (real, rather than test) data will be collected and validated so that the master files can be set up. **Security** procedures (see also Section 8.6) need to be tested, so that there is no unauthorized access and recovery is possible in case of failure. Once all this has been carried out, the system can be operated and the old one discontinued. If cutover to the new system is done 'overnight', then there could be problems associated with the new system. It is usually too risky an approach to cutover. Frequently, therefore, there is a period of parallel running, where old and new systems are run together, until there is complete confidence in the new system. Alternatively, parts of the new system can be implemented in turn, forming a phased implementation. If one part of the system is implemented 'to test the water' before the rest of the system is implemented, this is referred to as a pilot run. Acceptance testing comes to an end when the users feel assured that the new system is running satisfactorily and it is at this point that the new system becomes fully operational (or 'live').

6 Review and maintenance

The final stage in the system development process occurs once the system is operational. There are bound to be some changes necessary and some staff will be set aside for maintenance, which aims to ensure the continued efficient running of the system. Some of the changes will be due to changes in the organization or its environment, some to technological advances, and some to 'extras' added to the system at an agreed period following operational running. Inevitably, however, some maintenance work is associated with the correction of errors found since the system became operational.

At some stage there will also be a review of the system to ensure that it does conform to the requirements set out at the feasibility study stage, and the costs have not exceeded those predicted. The evaluation process might lead to an improvement in the way other systems are developed through the process of **organizational learning**.

As commented earlier, because there is frequently a divergence between the operational system and the requirements laid out in the feasibility study, there is sometimes a decision made to look again at the application and enhance it or even develop another new system to replace it. This could also occur for another reason. Changes in the application area could be such that the operational system is no longer appropriate and should be replaced. The SDLC then finishes and starts at the point where there is a recognition that needs are not being met efficiently and effectively and the feasibility of a replacement system is then considered and the life cycle begins again.

In the following sections we discuss the key concepts of methodologies, techniques, and tools in the particular context of the SDLC.

3.2 Methodology

There were many variants of the SDLC. The earliest in the UK was that proposed by the National Computing Centre (NCC) in the late 1960s. At the time, the NCC was sponsored by the UK government to provide a lead in computing for both government and private organizations. One of its functions, therefore, was to suggest good standards for developing computer applications. It had a great impact on the data processing community, particularly in the UK, and represented a typical methodology of the 1970s based on the SDLC. This methodology has been improved and altered since, so that a number of approaches used today can be regarded as a modern variant of this approach, the most obvious being SSADM (Section 21.1).

The use of a methodology improves the practice of information systems development. The attributes that we would expect of a methodology include:

1. A series of **phases** starting from the feasibility study through to review and maintenance. The phases are expected to be carried out as a sequential process. Each of these phases has subphases. The activities to be carried out in each of the subphases are usually spelt out clearly in the methodology documentation, usually found in manuals. The outputs (or 'deliverables') of each subphase are also detailed. Deliverables may include documents, plans, or computer programs.
2. A series of **techniques**, which include ways to evaluate the costs and benefits of different solutions and methods to formulate the detailed design necessary to develop computer applications, are also detailed. We illustrate some of these techniques in Section 3.3.
3. A series of **tools** to help the systems analysts in their work. By tools, we mean, in this book, software packages that aid some aspect of the approach. We discuss some of these tools in Section 3.4.
4. A **training** scheme so that all analysts and others new to their roles and responsibilities could adopt the standards suggested. Typically, there may be a training course offered that might take up to six weeks. A qualification might be offered.
5. A **philosophy**, perhaps implied rather than stated, which might be that 'computer systems are usually good solutions to organizational problems and processing'. We

suggest this because the assumption is that the inevitable consequence of using most methodologies is a new computer application to perform the required functions.

Of course, the NCC methodology itself became dated and was replaced by other methodologies, many of which were also based on the life cycle but incorporated the latest methods, techniques, and tools. Many more modern methodologies such as SSADM, Merise, and Yourdon Systems Methodology, which are described later in the book, are life cycle approaches.

For example, SSADM (Section 21.1) has the following stages:

- Stage 0 – feasibility;
- Stage 1 – investigation of current environment;
- Stage 2 – business systems options;
- Stage 3 – definition of requirements;
- Stage 4 – technical system options;
- Stage 5 – logical design;
- Stage 6 – physical design.

Although the terminology used for each stage seems different, we can see the progress through the life cycle from feasibility study, via investigation and requirements analysis, through to design. In fact, SSADM does not offer much specific advice regarding implementation, as this is seen as being particular to the application environment and the assumption is made that little general advice can be given. Each of these seven stages of SSADM are further detailed into several steps (the methodology documentation consists of several manuals).

Although we have argued that most methodologies follow a life cycle of sorts, some circumvent stages in the wish to develop applications quickly, such as James Martin's Rapid Application Development (Section 23.1), for example; whereas others do not assume such a logical step-by-step approach (Multiview, for example – Section 26.1). Yet others concentrate on only part of the life cycle, like Soft Systems Methodology (Section 25.1). Finally, some methodologies concentrate on particular views about the emphasis that should be placed when developing applications, such as ETHICS (Section 24.1), which emphasizes people aspects and participation.

3.3 Techniques

Most methodologies recommend a number of documentation aids to ensure that the investigation is thorough. The early methodologies might have included:

- various flowcharts, which might, for example, help the analyst trace the flow of documents through a department;
- an organization chart, showing the reporting structure of people in a company or department;
- manual document specifications, giving details of documents used in the manual system;

- grid charts, showing how different components of the system, such as people and machines, interact with each other;
- discussion records on which the notes taken at interviews could be recorded;
- file and record specifications describing, in the former case, all the data items in a record, including their names and descriptions, size, format, and possible range of values.

Many of the documentation techniques outlined above have been replaced. The most well-used techniques are probably now data flow diagramming (Section 12.1), which is important in specifying the process aspects of the system, and entity-relationship diagramming (Section 11.1), which details the data aspects. Data flow diagrams that describe the logic of the system or the physical implementation of the new design show:

- data flows into the system from the environment;
- data flows out of the system into the environment;
- processes that change the data within the system;
- data storage within the system;
- the boundary and scope of the system.

Entity-relationship diagrams, on the other hand, represent the data aspects. An entity is a data type, such as a student, lecturer, course, and classroom. These diagrams show how these entities relate to each other and they also show the detail about each entity, similar to that provided in a record specification form.

As we will see in Part IV, there are many techniques used in information systems development. Some are associated with particular methodologies, like rich pictures (Section 10.1) in Soft Systems Methodology, others are more generic, such as dataflow diagrams and entity-relationship diagrams, which are recommended in many methodologies.

3.4 Tools

When we talk of tools in this book, we are referring to software packages that aid aspects of information systems development. Most tools require significant computing power and good graphics capabilities. Most tools available now were not available in the early days, although there were project management tools (we discuss Microsoft Project in Section 18.4) and report-generating packages, such as RPGII. Users of RPGII would complete a series of forms specifying how the required report should be structured and what files contained the data necessary to be on the report. This has been incorporated as one aspect of modern toolsets (Chapter 19). Oracle, for example, will have an application generator as part of its toolset. Other tools support the production of documents, such as dataflow diagrams and entity-relationship diagrams. They include Visio, discussed in Section 18.3, but again this feature is included in many toolsets. As you will see in Part V, there are many different types of tool available, which ease much of the systems development process.

Probably the greatest benefit of using tools is that analysts and designers are not reluctant to change documents, because the change process is simple. Frequent manual redrawing is not satisfactory, because of the effort involved and the potential of introducing errors when redrawing. Without such drawing tools many a small change required by a user would not be incorporated into the documentation of the system.

3.5 Potential strengths of the SDLC

The SDLC has a number of features to commend it. Methodologies incorporating this view of applications development have been well tried and tested. The use of documentation standards in such methodologies helps to ensure that the specifications are complete, and that they are communicated to systems development staff, the users in the department, and the computer operations staff. It also ensures that these people are trained to use the system. The education of users on subjects such as the general use of computers is also recommended, and helps to dispel fears about the effects of computers. Following such a methodology also prevents, to some extent at least, missed cutover dates (the date when the system is due to become operational) and unexpectedly high costs and lower than expected benefits. At the end of each phase the technologists and the users have an opportunity to review progress. By dividing the development of a system into phases, each subdivided into more manageable tasks, along with the improved training and the techniques of communication offered, we have the opportunity for control of the applications development process.

3.6 Potential weaknesses of the SDLC

The criticisms of the systems development approach to applications development, or, to be more precise, of the way in which it was used include:

- failure to meet the needs of management;
- instability;
- inflexibility;
- user dissatisfaction;
- problems with documentation;
- lack of control;
- incomplete systems;
- application backlog;
- maintenance workload;
- problems with the 'ideal' approach;
- emphasis on 'hard' thinking;
- assumption of 'green-field' development.

We will discuss each of these in turn.

1 Failure to meet the needs of management

As can be seen in Figure 3.1, although systems developed by this approach often successfully deal with such operational processing as the various accounting routines (payroll, invoicing, etc.), the needs of middle management and top management can be ignored. Management information needs, such as that required when making decisions about where to locate a new

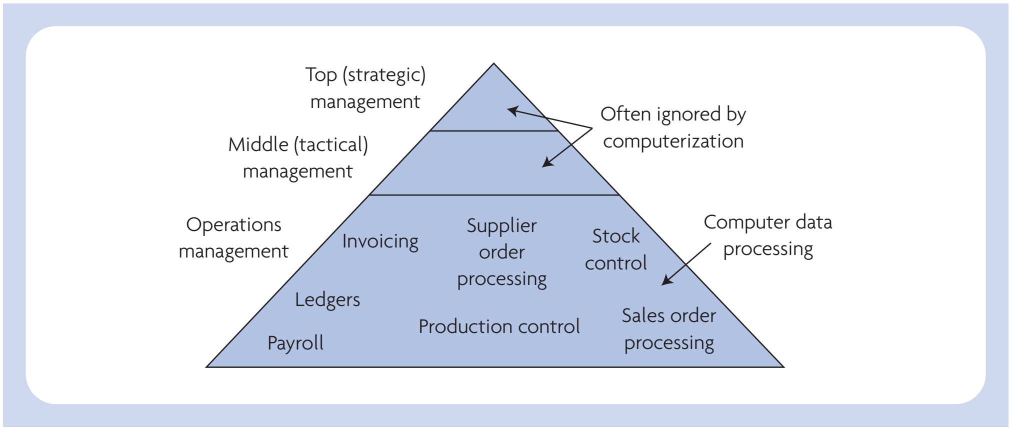


Figure 3.1: Failure to meet all the needs of management

factory, which products to stop selling, what sales or production targets to aim for and how sales can be increased, are neglected. Although some information may filter up to provide summary or exception reports, the computer is being used largely only for routine and repetitive tasks. Instead of meeting corporate objectives, computers are being used to help solve low-level operational tasks. In general, the danger is that such systems are rather limited in their scope and rather unambitious.

2 Models of processes are unstable

The conventional SDLC approach attempts to improve the way that the processes in businesses are carried out. However, businesses do change, and processes need to change frequently to adapt to new circumstances in the business environment. Because these computer systems model processes, they have to be modified or rewritten frequently. It could be said therefore that computer systems, which are to some extent 'models' of processes, are unstable because the real-world processes themselves are unstable.

3 Output-driven design leads to inflexibility

The outputs that the system is meant to produce are usually decided very early in the systems development process. Design is 'output driven' (see Figure 3.2) in that, once the output is agreed, the inputs required, the file contents, and the processes involved to convert the inputs to the outputs can all be designed. However, changes to required outputs are frequent and, because the system has been designed from the outputs backwards, changes in requirements usually necessitate a very large change to the system design and therefore cause either a delay in the implementation schedule or are left undone, leading to an unsatisfactory and inappropriate system.

4 User dissatisfaction

This is often a feature of many computer systems. Sometimes systems are rejected as soon as they are implemented. It may well be only at this time that the users see the

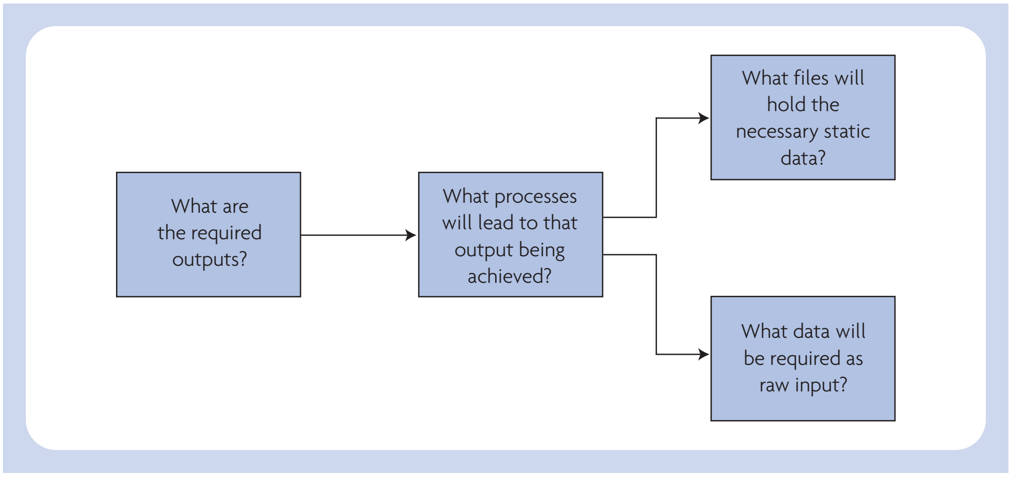

What are the required outputs?
What processes will lead to that output being achieved?
What files will hold the necessary static data?
What data will be required as raw input?

Figure 3.2: Design is output-driven

repercussions of their 'decisions'. These decisions have been assumed by the analysts to be firm ones and, as computer systems can prove inflexible, it is normally very difficult to incorporate changes in requirements once the systems development is under way. Many companies expect users to 'sign off' their requirements at an early stage when they do not have the information to agree the exact requirements of the system. The users sign documentation completed by computer-oriented people. The documents may not be designed for the users. On the contrary, they are designed for the systems analysts, operations staff, and programming staff who are involved in developing the system. Users cannot be expected to be familiar with the technology and its full potential, and therefore find it difficult to contribute to the debate to produce better systems. The period between 'sign-off' and implementation tends to be one where the users are very uncertain about the outcome and they are not involved in the development process and therefore may lose their commitment to the system. They might have become disillusioned with computer systems and, as a consequence, fail to co-operate with the systems development staff. Systems people, as they see the situation, talk about user staff as being awkward or unable to make a decision. Users may first see the system only on implementation and find it inappropriate. This alienation between technologists and users has even been known to lead to users developing their own informal systems which are used alongside the computer system, ultimately causing the latter to be superfluous.

5 Problems with documentation

We have suggested that one of the benefits of the methodology is the standardization and documentation that methodologies use. Yet this is not ideal. The orientation of the documentation is frequently toward the computer person and not the user, and this represents a potential source of problems. The main purpose of documentation is that of communication, and a technically oriented document is not ideal.

6 Lack of control

Despite the methodological approach enabling estimates of the time, people, and other resources needed, these estimates can be unreliable because of the complexity of some phases and the inexperience of the estimators. Computer people have been seen as unreliable and some disenchantment has ensued in relation to computer applications.

7 Incomplete systems

Computers are particularly good at processing a large amount of data speedily. They excel where the processing is the same for all items: where the processing is structured, stable, and routine. This often means that the unusual (exceptional conditions) is frequently ignored in the computer system. They are too expensive to cater for. If they are diagnosed in the system investigation stage, then clerical staff are often assigned to deal with these exceptions. Frequently they are not diagnosed, the exceptions being ignored or forgotten, and the system is falsely believed to be complete. These exceptions cause problems early on in the operational life of the system. These problems mark a particular technical failure, but they also indicate a systems failure, a failure to analyse and design the system properly.

8 Application backlog

A further problem is the application backlog. There may well be a number of systems waiting to be developed. The users may have to wait some years before the development process can get under way for any proposed system. The process to develop an information system will itself take many months and frequently years from feasibility study to implementation. Users may also postpone requests for systems because they know it is not worth doing because of the backlog. This phenomenon is referred to as the invisible backlog.

9 Maintenance workload

The temptation, then, is toward 'quick and dirty' solutions. The deadline for cutover may seem sacrosanct. It might be politically expedient to patch over poor design rather than spend time on good design. This is one of the factors that has led to the great problem of keeping operational systems going. With many firms, the effort given to maintenance can be as high as 60–80 per cent of the total workload. With so many resources being devoted to maintenance, which often takes priority, it is understandable that there is a long queue of applications in the pipeline. The users are discouraged by such delay in developing and implementing 'their' applications because of the necessity to maintain the legacy systems (see Section 8.1).

10 Problems with the 'ideal' approach

The SDLC model assumes a step-by-step, top-down process. This is somewhat naive. It is inevitably necessary to carry out a series of iterations when, for example, new requirements are discovered or subphases might prove unnecessary. In other words, information systems development is an iterative process, whatever textbooks say. The political dimension where, for example, people have their own 'agenda' transcends the rationale of any methodology. Users and analysts will need to interpret the methodology, and its techniques and tools, to be relevant to a particular problem situation. It also assumes a tailor-made rather than packaged solution, usually for a medium- or large-scale application using mainframe computers. With the widespread use of PC systems, such an approach may be inappropriate and unwieldy.

11 Emphasis on 'hard' thinking

The SDLC approach may make a number of simplistic assumptions. It assumes that there are 'facts' that only need to be investigated and identified; it assumes that there can be a 'best solution' identified that will 'solve the problem'; it assumes that this 'ideal solution' can be easily engineered by following a step-by-step methodology; and it assumes that the techniques offered will analyse and design all that needs to be done. But we are not engineering a simple mechanical object. The world of information systems is concerned with organizations and people as much as technology. The situations encountered are often ambiguous, issue-laden, messy, and problematical, and an alternative approach might well be more appropriate in these difficult but common situations.

12 Assumption of 'green-field' development

The SDLC has been criticized for embodying the assumption that all systems developed are new and that developers begin with a 'green field', that is, there are no existing systems that have to be taken into consideration. This may have been true when the first computerized systems replaced manual systems, but it is no longer the case. New developments are not nicely separated from each other and any new development work is more likely to be an incremental development of an existing system than a complete replacement. It is also likely to have to be integrated with many existing systems in an organization, typically having been developed at different times, with different developers, on different architectures and platforms. This makes development a much more messy activity than would otherwise be the case. This assumption is not confined to the SDLC, as many methodologies also implicitly make this assumption. One that specifically addresses the complexity of existing systems is Renaissance (see Section 25.5). This issue is also addressed elsewhere, for example, in evolutionary development (Section 7.1) and legacy systems (Section 8.1).

The above criticisms of the SDLC should be regarded as 'potential' criticisms, as many organizations using the approach do not fall into all or even most of the potential traps. However, this book concerns itself with improving information systems development by adapting the SDLC or using alternative approaches to developing information systems.

In Part III, we will look at themes regarding differing views about information systems development. They can be seen as ways in which people in the field have reacted to the potential problems of the SDLC. This has resulted in many views about what a methodology should emphasize: people aspects, planning aspects, automation . . . there are many such views. To some extent these can be seen as representing different philosophies, for example, a people-oriented approach might be seen as being based on the philosophy that people have a right to design their organizational environment. But these themes also reflect greater possibilities, for example, software tools that can both make the process easier and faster, and also make some approaches possible, which aim toward automation of the development process. Similarly, the craft of systems analysis has been improved greatly through the use of alternative techniques.

3.7 Conclusion

Before we consider approaches to systems analysis, which represent advances on the traditional SDLC approach, the reader should stop and consider that many systems developed today are still done using no real methodology at all. This is particularly true in organizations using PCs, perhaps developing websites as their first experience of computing. The SDLC has many advantages over trial and error.

The SDLC is being used successfully. In many respects, there is nothing intrinsically wrong with the SDLC. Much depends on the way in which it is used. There must be sufficient resources assigned to the process; it needs to be well managed and controlled so that any deviation from the plan is noticed and dealt with; it should be seen not as a rigid process but as a flexible and iterative one; the feasibility study needs to be seen as a way of exploring alternatives rather than as a way of advancing a limited point of view; and systems development should not be seen as a purely technological process but one involving all users and developers, indeed the organization as a whole. However, many of the alternative approaches address some of the potential weaknesses suggested above. This may mean improving one or more of the phases, including the use of an alternative technique or a new software tool.

We also do not wish to assert that any of the alternative methodologies represents a panacea. Major concerns in computing remain, for example:

- meeting project deadlines;
- system maintenance;
- staff recruitment and retention;
- user dissatisfaction;
- changing requirements.

Notwithstanding these continuing pessimistic trends, there are many successful information systems, some developed using the basic SDLC approach. In Part III we will look at the various themes in information systems development methodologies that are seen as developments or alternatives to the basic approach discussed in this chapter. These alternative approaches usually address one or more of the potential criticisms of the SDLC approach discussed in the previous section.

Summary

- The SDLC has had an enormous influence as a general approach to develop information systems. Although there are many variants, it has the following basic structure: feasibility study; system investigation; systems analysis; systems design; implementation; review and maintenance.
- An information systems development methodology is likely to include: a series of phases with subphases, each having expected outputs (or 'deliverables'); a series of techniques; a series of tools; a training scheme and some underlying philosophy.
- Although the SDLC has many strengths and is still used today, it also has many potential weaknesses, which has led to alternative methodologies, techniques, and tools being available.

Questions

1. Distinguish between techniques, tools, and methodologies in the context of the SDLC.
2. Discuss the strengths and the limitations of the traditional systems development life cycle.
3. Would you adopt the SDLC for developing a small application on a PC? Would you modify it in any way? Give reasons for your answer.

Further reading

Avison, D.E. and Shah, H.U. (1995) *The Information Systems Development Cycle: A First Course in Information Systems*, McGraw-Hill, Maidenhead, UK. Avison and Shah describe an up-to-date approach modelled on the SDLC. It can be regarded as a prequel to this present book.