Using System Development Methods

BUTLER COX FOUNDATION

Research Report 57, June 1987

BUTLER COX
FOUNDATION

Using System Development Methods

Research Report 57, June 1987

Butler Cox & Partners Limited

LONDON
AMSTERDAM MUNICH NEW YORK PARIS

Published by Butler Cox & Partners Limited
Butler Cox House
12 Bloomsbury Square
London WC1A 2LL
England

Availability of reports
Members of the Butler Cox Foundation receive three copies of each report upon publication; additional copies and copies of earlier reports may be purchased by members from Butler Cox.

Photoset and printed in Great Britain by Flexiprint Ltd., Lancing, Sussex.

BUTLER COX FOUNDATION

Using System Development Methods

Research Report 57, June 1987

Contents

1 The need for system development methods 1

2 Methods, tools, and techniques, and the system development process 3
The system development process 3
More than one development process is required 6
A working definition of system development techniques, methods, and tools 6
Summary 7

3 Proprietary methods by themselves are not enough 9
Categories of method 9
No proprietary methods can be used for all types of development process 10
Proprietary methods do not provide full support for all development phases 12
Development methods need to be supported by appropriate tools 14
Summary 18

4 Selecting appropriate methods and tools 19
Identify the development processes to be used 20
Decide where proprietary methods can best be used 20
Choose an appropriate method 20
Criteria for selecting a method 23
Choose tools to support the methods 25

5 Managing the implementation of methods 26
Obtain senior management consent 26
Consider the organisation of the systems department 26
Gain experience in using the new methods 27
Implement a new method as a distinct project 27

6 Monitoring the payback from system development methods 29
Payback achieved depends on the type of method 30
Improved quality of systems 30
Improved control over the development process 31
Do not expect productivity gains from methods alone 32
Implement advanced tools to support proprietary methods 32

Report conclusion 33

BUTLER COX FOUNDATION

Management Summary

A Management Summary of this report has been published separately and distributed to all Foundation members. Additional copies of the Management Summary are available from Butler Cox

Chapter 1
The need for system development methods

Application system development is inherently a highly complex and difficult task. In many organisations, the system development group typically develops several kinds of applications. These may include process-control, transaction-processing, and decision-support systems. There may also be systems based on novel technology such as expert system shells. These systems may be developed for a variety of computers, in a number of languages, and they may vary greatly in size. The level of experience of the users and developers of these application systems varies, as may the level of business and development risk associated with each system and the urgency with which they are required. Because of the complexity and difficulties of system development, and the different types of system that have to be developed, several different system development processes may be used — one for transaction-processing systems, another for decision-support systems, and so on.

Moreover, the demand for new systems still exceeds the capabilities of most organisations to deliver them, and many systems being developed now are more complex than in the past. The result is that, in many organisations, the systems development department continues to be perceived as delivering systems much later than was planned, and at much higher cost. And often the systems that are delivered do not match their users' needs.

Most system development managers recognise that there is room for improvement in the way that systems are built. They realise that, all too often, they are providing systems that are lacking in quality, delivered late, and cost more than was budgeted. They would like the development process to be more manageable and less dependent on the skills of the individual analysts and programmers — experts who are in short supply and expensive to train. In other words, they would like a well-defined systematic process for developing systems.

In an attempt to solve these problems, many organisations have tried to standardise their development processes by using a proprietary development method supported by appropriate tools. A wide range of such methods and tools are now in use, with each one promoted by its supplier as the solution to all development problems. Many system development managers are confused by the plethora of products and the competing claims of the various suppliers.

In setting out to research this topic, we were asked by many Foundation members to provide advice on choosing a single system development method that could be used to solve all their organisation's development problems. The underlying belief was that system development methods had now reached the stage where it was possible to do this.

However, our research showed that user organisations are discovering that a method promoted as an all-embracing solution to system development is actually applicable only to specific phases of the development process. This means either that additional methods have to be purchased to cover the other phases, or that in-house procedures have to be used as well. And many organisations are discovering that proprietary methods often do not provide the management procedures necessary to ensure the success of development projects.

Furthermore, user organisations often are not making the best use of proprietary development methods. In some cases, a method has been abandoned because the analysts and programmers were not provided with the development tools that would have allowed them to use the method effectively. This is in part due to the fact that many people (users and suppliers) are confused about what a system development method is and how it relates to development tools and development techniques. Users are also unclear about the relative benefits to be gained from using methods and tools. This confusion is compounded by the exaggerated claims that some suppliers of methods and tools make for their products.

It is also costly to implement new methods and the support they require, and considerable management effort is required to ensure that a method is implemented properly. The hoped-for benefits are not always realised, and some of the benefits are difficult to quantify in any case. Hence it is hard to justify an investment in new methods.

Because of these findings, we soon reached the conclusion that it is not possible for an organisation to expect a single proprietary method to solve all development problems. Nor is this situation likely to change in the foreseeable future. The focus of our research therefore shifted to determining how Foundation members could make the best use of the proprietary system development methods that are currently available.

The conclusion we reached is that each organisation needs to use proprietary system development methods selectively to attack those development activities that cause the most problems. Different proprietary methods might be used for project management, for systems analysis and design, and for programming. Alternatively, a proprietary method might be used only for the programming phase, with in-house standards being used for the rest of the development process.

Before deciding whether to implement a proprietary development method, it is first necessary to understand clearly the relationships between methods, techniques, and tools, and their respective contributions to the systems development process. We cast some light on this confused area in Chapter 2.

In Chapter 3, we explain why a single proprietary method is not a sufficient solution to an organisation's development problems. No method covers all the phases of the development process, and even those phases that are covered may be incompletely covered. Furthermore, a particular method may be suitable for one type of development process but not for others. Chapter 3 also shows that installing only one method by itself may actually be counterproductive. It is necessary to install development tools as well.

Chapter 4 provides advice about how to select proprietary system development methods and the tools that will be required to support them. Many of the alternative methods provide similar technical facilities, and two important selection criteria are the stability of the supplier and the support provided by the supplier.

One of the keys to implementing a proprietary method successfully is careful management of its introduction. Chapter 5 provides advice on how to do this.

Finally, once a new method has been implemented, it is necessary to monitor the benefits being achieved by its use. Chapter 6 identifies the potential benefits that should be monitored.

The report is based mainly on research carried out at the end of 1986 and the beginning of 1987. Some seventy interviews were conducted both with suppliers and with users of proprietary methods throughout Europe. We also drew on the opinions of a number of experts on this topic.

Apart from this specific research, we also referred to our other related research and extensive consultancy experience in this area, including that for Foundation Report 47 — The Effective Use of System Building Tools.

The research for this report was led by Mary Cockcroft, a principal consultant with Butler Cox. Mary has carried out a wide range of consultancy assignments advising large organisations about system development strategies and the use of development methods. She was assisted by Rob Moreton, an associate consultant with Butler Cox who has extensive experience of the theory and application of development techniques; David Flint, a principal consultant with Butler Cox and the author of Foundation Report 47; and Simon Forge, a senior consultant with Butler Cox's Paris office.

Chapter 2
Methods, tools, and techniques, and the system development process

There is a considerable amount of confusion, among user organisations and suppliers of development methods and tools, about the meaning of the terms 'development method', 'development technique', and 'development tool'. This confusion is due largely to a misconception about how a proprietary development method helps with the system development process. Contrary to popular belief, most proprietary methods help with only part of the total process. Indeed, in some cases, it may be necessary to use several proprietary methods, each one covering a different part of the life cycle. Thus, before we can describe the relationships between methods, tools, and techniques, we first need to explain what we mean by the system development process — the series of activities that encompasses the whole of the systems life cycle.

THE SYSTEM DEVELOPMENT PROCESS

The system development life cycle typically begins with a survey or feasibility study and ends with an operational system that is then modified and maintained until the end of its useful life. The need to manage systems throughout the whole of their life cycle has been recognised for a decade or more, but, surprisingly, we found some major organisations with multimillion-dollar investments in systems that were only now introducing the life-cycle concept. Traditionally, they had formed project teams of analysts and programmers, provided the project manager with a copy of the standards manual (which typically specified the programming standards to be used), and left the team to get on with the development of the system. These companies now recognise that a more methodical approach to the whole system development process could improve the quality, cost, and timeliness of development.

Our first concept, then, is that of the system development process. We use this term to describe all the phases and activities that make up the complete life cycle of a system. Depending on the size and complexity of the application being developed, the development process may take anything from a few days to several years. There are essentially two major types of system development process:

- The conventional linear process, where progress is achieved by proceeding in a linear fashion through each successive phase of the life cycle.
- The iterative process, where several passes are made through one or more of the life-cycle phases, with additional functionality and detail being added with each pass. The iterative process usually relies on the use of prototyping at the requirements-definition phase.

Some organisations now use both types of system development process, selecting the one most suited to the application to be developed. We have also identified six other alternative types of development process, which are variants of the two main types. Each development process is described in more detail below.

THE CONVENTIONAL DEVELOPMENT PROCESS

The conventional development process has been used for many years for the development of commercial and business applications. Typically, the work is subdivided into well-defined steps or phases, with the workflow being controlled and monitored by formal project-management techniques. More recently, proprietary system development methods have been used to standardise the tasks carried out during one or more of the life-cycle phases.

THE ITERATIVE DEVELOPMENT PROCESS

An iterative development process is more appropriate for applications where the users' requirements are less easy to specify and where the scale of the application is small enough to allow a prototype to be built and revised quickly using advanced system-building tools such as Natural, Focus, Mapper, or Linc.

It is more difficult to use formal project-planning and control techniques with the iterative development process because the number of iterations that will be required cannot generally be predicted in advance. In practice, iterative development is better managed as a sequence of small projects, with each project typically lasting for between six and nine months.

ALTERNATIVES TO CONVENTIONAL AND ITERATIVE DEVELOPMENT

In practice, large organisations will need to develop systems in ways that do not correspond exactly with the conventional development process or the iterative development process. The development process used in practice will depend on the nature and size of the application, its urgency, and whether it is to be developed by data processing staff or by end users. Figure 2.1 shows the six main alternatives to the conventional and iterative development processes. They have the characteristics described below, and they require different facilities in the methods that could be used to support them.

Small-systems development process

The small-systems development process is appropriate for new small systems and for small enhancements to existing systems. Typically, the development takes less than nine months' elapsed time and requires no more than about two or three man-years of effort. The small-systems development process covers the same range of applications as the conventional development process.

A small-system development process is typically a variant of the conventional development process, but, because the timescale of development is shorter and there are fewer development staff working on the project, less stringent project-management techniques are required. Such development commonly uses structured analysis and design, but prototypes may also be used for requirements specification and as the basis of the implemented system. Advanced system-building tools such as fourth-generation languages can be used for small-systems development.

Some organisations now deliberately subdivide large complex systems into several subsystems, each of which is implemented using the small-systems development process. The advantages are that the management of the overall project is simplified and subsets of the total system are delivered earlier than they would be otherwise.

Accelerated development process

Accelerated development is used to build a working system as quickly as possible in situations where the operational performance of the completed system is not a major concern. This development process is based on the use of advanced system-building tools such as Natural, Focus, Mapper, and Linc.

In accelerated development, the tools are used to build prototypes for requirements analysis and are also used to construct the final system. The use of prototyping and the high productivity in the construction phase allows smaller teams and shorter timescales, so that, relative to conventional development, documentation and project-management requirements can be reduced. Accelerated development requires fewer, but more skilful, development staff. Accelerated development can, given

Figure 2.1 Basic system development processes and alternatives

Development process	*Nature and size of application*	*Urgency to develop the system*	*System developer*
Basic processes			
Conventional (linear)	Medium-to-large Conventional data processing	Normal	Systems department
Iterative	Small-to-medium Difficult to predefine user requirements	Normal	Systems department
Alternative processes			
Small-systems	Small Conventional data processing	Normal	Systems department
Accelerated	Small Conventional data processing Operational performance not a concern	Urgent	Systems department
Application-package	Standard data processing applications	Normal/urgent	Systems department
End-user	Very small Department or user-specific	Normal/urgent	End user
Specialist-applications	Realtime Process control	Normal	Specialist department/ systems department
Emergency	Any	Extreme urgency	Systems department

good staff and tools, be applied to data processing and decision-support systems requiring up to five, or even ten, man-years of development effort. It may be used as an alternative to the small-systems development process (except for enhancements), or even for systems that would otherwise require the conventional process.

Application-package development process
For many types of application, particularly in the financial and accounting areas, packages providing a large proportion of the functionality required by the users already exist. As packages improve in quality and coverage and become less expensive than customised in-house development for a wider variety of applications, the need to be able to buy in packages and customise them will increase. A development process is therefore needed that can identify, select, adapt, and implement the packages. Such a process usually incorporates the following phases:

— Analysis of requirements.

— Review of the packages available in the market, and selection of a shortlist.

— Comparison of the application requirements with the deliverables of each shortlisted package and selection of a package.

— Tailoring of the package to match the application requirements more closely.

— Implementation of the package.

The application-package development process therefore includes some phases not required in conventional development, but it does impose some additional constraints. For example, the tools available for tailoring the package may be specific to the actual package chosen.

End-user development process
Users should be involved in all application system developments. However, some applications can be developed by the end users themselves. These applications provide limited functionality and are usually very small systems, designed to meet an individual or departmental requirement.

For users, system development is just one of their responsibilities, and they neither need nor can they be expected to be proficient in the use of a comprehensive development process intended for full-time professional system development staff. But the systems department can provide the users with guidance by:

— Identifying appropriate system-building tools that can easily be used by end users and that are consistent with the organisation's systems architecture.

— Training users in the use of simple analysis techniques and the system-building tools.

— Setting up a central support group (for example, an information centre) that can provide assistance and consultancy effort to end users when required.

Specialist-applications development process
In addition to conventional data processing and end-user computing applications, there are also specialist applications such as 'embedded' systems, process-control, and realtime applications.

The development process used for these applications is similar to a conventional linear process, using formal project-management techniques at each development phase, as well as structured-analysis, design, and programming techniques.

However, the major differences between these specialist applications and data processing applications are that:

— The level of 'correctness' and reliability required in these systems is high. Correctness is the extent to which the system satisfies its specification, and reliability is the extent to which a system can be expected to perform its intended function with the required precision.

— The design of the application is oriented towards activities rather than data, and there are time constraints on system operation in realtime systems.

These differences, and the fact that the users of specialist applications are often engineers, mean that the use of 'formal' methods for specification and design is much more prevalent in realtime development. (Formal methods are described in more detail in Chapter 3 on pages 11 and 12.)

In addition, the tools used for these specialist applications are different from those used in traditional data processing. Operational performance is a major concern in these systems, and the tools used reflect this concern. As a consequence, little use to date has been made of advanced system-building tools for implementing specialist applications, apart from the requirements-definition phase.

If specialist applications are not a routine part of their activity, most organisations subcontract this type of development to software or systems houses. Other organisations form separate development teams dedicated to this kind of work, and these teams use a development process that is different from that used for data processing applications.

Emergency development process
Sometimes, it will be necessary to develop an application in a very short time. In some cases, it will be possible to use an accelerated development process, but even this may not be sufficient to meet the urgent timescales required. With such applications, time is of the essence, and the cost of development is a secondary concern.

The keys to success in developing emergency applications are to minimise the technical risks, avoid distractions, and ignore costs. The methods and tools used should therefore be those with which the team are most familiar (provided that they are adequate). The development process for emergency applications will therefore vary according to the requirements of the particular application. The system development manager and the project manager should decide between them which of the available methods and tools are most applicable.

They may decide, for example, that they will use an iterative process without strict project-management techniques. The development team might be provided with an advanced integrated project-support environment together with advanced system-building tools.

MORE THAN ONE DEVELOPMENT PROCESS IS REQUIRED

From the description above of the different types of development process, it is evident that an organisation is unlikely to find a single development process that is adequate for all its needs. Some organisations try to use a standardised conventional process for all types of application, and this has caused problems. A common difficulty is caused by spending too much time on managing small projects, because the standard process requires all phases of the life cycle to be recognised.

Several organisations reported that delays in developing small systems were a major source of user dissatisfaction. Some of these organisations reported that development staff begin either not to use the standards for the conventional development process, or to pay only token attention to their use, even in those circumstances where they are necessary. On the other hand, the standards used by some organisations mean that they do not spend enough time on planning and managing large projects. The result is that costs increase and timescales slip on large projects.

Other organisations attempt to use a development process based on a proprietary method for inappropriate applications. For example, one company had used the Vienna Development Method (VDM) for a decision-support system that involved senior user management during the analysis phase. This company found that the very formal mathematical VDM method was inappropriate for this type of application because the users did not understand the products of the analysis phase (a stream of mathematical symbols).

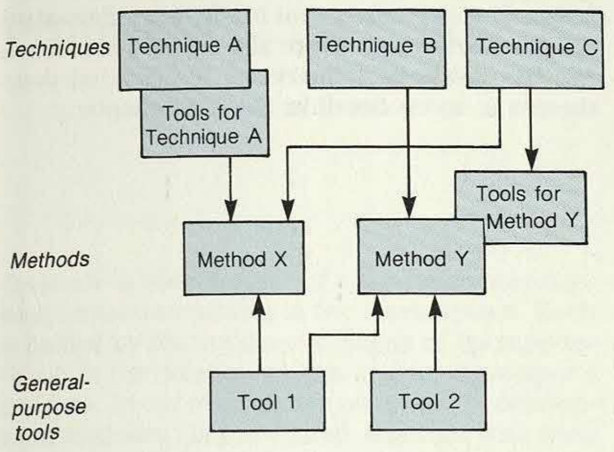
A WORKING DEFINITION OF SYSTEM DEVELOPMENT TECHNIQUES, METHODS, AND TOOLS

Our research showed that the terms 'system development techniques', 'system development methods', and 'system development tools' are used to mean different things by different people. Here we define these terms as we use them in this report. We believe that the definitions help to clarify this confused area and make it easier to understand some of the misconceptions that users and suppliers have about the products they use and supply. The definitions are summarised in Figure 2.2, and the interrelationships between techniques, methods, and tools are shown diagrammatically in Figure 2.3.

Figure 2.2 Definitions of system development techniques, methods, and tools

Techniques	The rigorous procedures on which system development is based. Often, techniques are developed by academics and made available to all through research papers. Examples of techniques include data analysis, functional decomposition, entity life history, prototyping, and structured programming. Most of these techniques are concerned with the how-to-do-it, rather than the what-to-do, aspects of systems development.
Methods	A system development method is a way of implementing in practice the ideas embodied in a system development technique. The same technique may form the basis of several competing methods. A method specifies either how to carry out a series of activities, or the procedures for determining which activities have to be carried out, or a combination of both. Different types of method cover different phases of the development life cycle.
Tools	Development tools automate the activities within a development method. Indeed, without such tools, many methods are very difficult to use in practice. Some tools are specific to a particular method; others to a particular technique. Other tools are generic in nature because they are independent of the method being used.

Figure 2.3 **Interrelationships between system development techniques, methods, and tools**



DEVELOPMENT TECHNIQUES

System development techniques are the rigorous procedures on which system development methods are based. In general, techniques are nonproprietary because they are developed by academics and are made available to the world at large through research papers. Examples of system development techniques include data analysis, functional decomposition, entity life history, prototyping, and structured programming. Sometimes a technique is originated by a consultancy firm and is available only from them as a packaged proprietary method. Most techniques are concerned with the 'how-to-do-it' (as opposed to the 'what-to-do') aspects of systems development.

DEVELOPMENT METHODS

System development techniques are commercialised as proprietary system development methods. In other words, a system development method is a way of implementing in practice the ideas embodied in a system development technique. (Sometimes a proprietary method may, in fact, be based on several techniques.) Suppliers may take the same basic technique and 'package' it in different ways, and today, many of the best-known proprietary methods are based on the same underlying techniques. For example:

- Yourdon's method is based on data analysis and functional decomposition.
- LSDM is based on data analysis, functional decomposition, the entity life-history concept, the life-cycle concept, and phased project-management techniques.
- Prism is based on the life-cycle concept and phased project-management techniques.

A system development method specifies either how to carry out a series of activities, or the procedures for determining which activities have to be carried out, or a combination of both. The purchaser of a proprietary method receives a procedures manual describing the activities that have to be carried out and a series of standard forms to be completed as specified in the manual. Until recently, methods' suppliers did not provide any development tools for use with their method. It was up to the user organisation to work out how best to carry out the activities specified in the procedures manual.

DEVELOPMENT TOOLS

More recently, development tools for automating the activities within a development method have become available. Indeed, many suppliers have recognised that without such tools, their methods are very difficult to use in practice, and they now provide tools designed specifically for use with their methods. An example is IEF (Information Engineering Facility), which is designed specifically for use with James Martin's Information Engineering method. Another is the Yourdon workbench product, which is designed for use with the variants of the data-analysis and functional-decomposition techniques used by the Yourdon development method.

Other development tools are generic in nature — that is, they are not designed for use with a specific method, but for a range of methods based on particular techniques. An example of such a product is Index Technology's Excelerator (an analyst's workbench product), which is designed for use with methods that use data-analysis and functional-decomposition techniques.

Some generic development tools can be used at different phases of the systems life cycle and are independent of the development methods being used. They can range in complexity from flowchart templates to sophisticated integrated project-support environments (IPSEs). Sometimes, generic development tools are software-based (system-building tools, for example, which were the subject of Foundation Report 47 — The Effective Use of System Building Tools), or a combination of hardware and software. Philips Maestro is an example of a development tool based on a combination of hardware and software.

SUMMARY

A particular development method may be based on several development techniques, and the same technique may be used by different methods. Development tools may be specific to a particular method or technique, or they may be generic tools that can be used with a range of methods. Proprietary development methods will therefore differ from each other in the extent to which they cover the total development process and in the extent to which they cover the what-to-do and how-to-do-it aspects of the phases they do cover.

The definitions of system development technique, method, and tool presented above should make it clear that one single proprietary development method is unlikely to be sufficient for all of the development processes that are likely to be used. It should also be clear that methods by themselves are insufficient; tools are also needed in order to use the methods effectively. We develop these themes in more detail in the next chapter.

Chapter 3

Proprietary methods by themselves are not enough

There are a wide variety of proprietary development methods available in the marketplace. Each is backed by the vociferous claims of its supplier that it is the solution to the system developer's problems. In our research we reviewed 21 development methods. They are listed, together with their suppliers, in Figure 3.1.

Figure 3.1 System development methods reviewed during the research

Method	*Supplier*
Management methods	
BIS Modus	BIS
MCP	Nonproprietary; refer to M Gedin, RATP, Paris
Method 1	Arthur Andersen & Co
Prism	Hoskyns Group Ltd
SDM (System Development Method)	Cap Gemini Sogeti
SPDM (Small Project Development Method)	Hoskyns Group Ltd
Single-phase methods	
Core	Systems Designers plc, British Aerospace plc
Ethics	Nonproprietary; refer to Prof E Mumford, Manchester Business School
JSP (Jackson Structured Programming)	Michael Jackson Systems Ltd
Slim (Sofware Life Cycle Management)	QSM Europe Ltd
Multiphase methods	
Delta	Delta Software Tools Ltd
IDA (Interactive Design Approach)	METSI
JSD (Jackson System Development)	Michael Jackson Systems Ltd
Mascot (Modular Approach to Software Construction, Operation and Test)	Software Sciences Ltd, Ferranti Computer Systems Ltd, and others
Merise	Cap Gemini Sogeti, Sema-Matra
SADT (Structured Analysis and Design Techniques)	Softech Computers
Soft Systems	Nonproprietary; refer to Prof Checkland at Lancaster University
Yourdon	Yourdon Europe
Integrated methods	
Information Engineering	James Martin Associates
LSDM/SSADM } LSDM Fastpath }	LBMS (Learmonth & Burchett Management Systems)
Stradis	P-E Consultancy Services

CATEGORIES OF METHOD

Development methods differ from each other in the functionality they offer in two main respects:

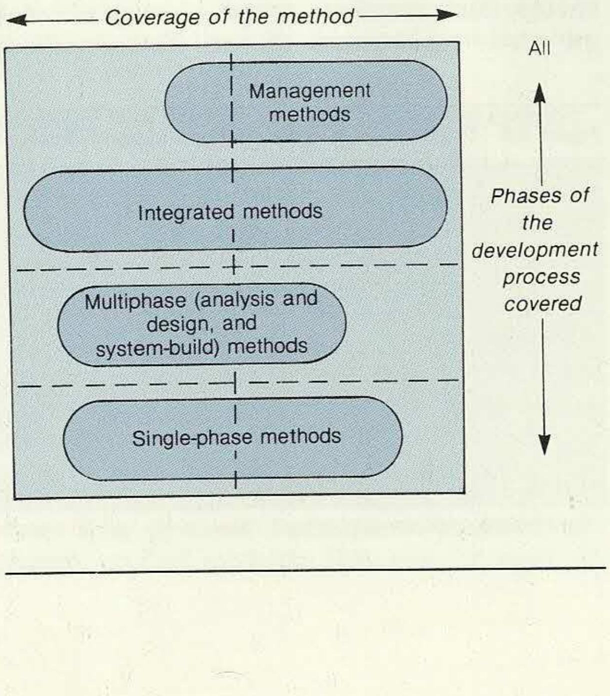
- The extent to which they cover the phases of the development process ('what to do').
- The extent to which they define how the development phases are to be executed ('how to do it') as well as what is to be done.

Figure 3.2 shows the four distinct categories of system development method (management methods, single-phase development methods, multiphase methods covering two or three consecutive development phases, and integrated methods) and how they differ with respect to these two parameters. (In addition, some methods, VDM for example, are designed to be used with particular types of application, such as realtime process-control systems.)

MANAGEMENT METHODS

Some proprietary methods specify the various phases of the development process, stating the

Figure 3.2 Categories of system development method



Coverage of the method
All
Management methods
Integrated methods
Phases of the development process covered
Multiphase (analysis and design, and system-build) methods
Single-phase methods

objectives of each phase together with what is to be done and the deliverables. Typically, these methods are document-orientated and tend to rely on bureaucratic procedures. They usually cover most phases of the development process, helping primarily with the management of the project rather than the execution of each phase. They can be called management methods.

Management methods originate mainly from systems consultancies that have developed the methods as an aid for their own work. Some examples are Method 1, Prism, and MCP.

SINGLE-PHASE DEVELOPMENT METHODS

At the other extreme of the methods spectrum are those that address only one phase of the development process. Their emphasis is on how the phase is to be undertaken, although they do also provide some guidance on what is to be done. Examples include QSM's Slim (for the project-planning and estimation phases), and Jackson's Structured Programming.

MULTIPHASE DEVELOPMENT METHODS

Some methods address more than one phase of the development process but concentrate on the analysis and design phases, the core of the development life cycle. Others (the system-build methods) focus on the programming and testing phases. Like the single-phase development methods, they too define how the phase is to be undertaken, providing limited guidance as to what needs to be done. Typical examples of these methods are the Yourdon development method, SADT (Structured Analysis and Design Techniques), and Merise.

INTEGRATED METHODS

Finally, there are methods that attempt to cover the whole development process. These methods define both what is to be done and how it is to be accomplished. They are integrated methods both in the sense of providing project-management and development methods, and in the sense of linking the techniques, methods, and tools used in the different phases of the development process.

They are the most complex and comprehensive form of method. Examples are James Martin's Information Engineering and LSDM/SSADM. (LSDM and SSADM are, in effect, the same product. SSADM is the version of LSDM that is used in government installations in the United Kingdom.)

NO PROPRIETARY METHODS CAN BE USED FOR ALL TYPES OF DEVELOPMENT PROCESS

All of the proprietary methods we reviewed can be used with the conventional linear development process, although none of them covered every phase. However, none of them could be used with all types of development process. At the best, most of the methods provided limited support for only two or three development processes.

Many of the methods were not suitable for specialist (realtime and process-control) systems because of their emphasis on data rather than activity or event analysis, which is critical to many such systems. Specialist systems were, however, catered for by methods developed for their particular needs.

Several methods recognise the need to cater for small as well as medium-to-large systems development. But the other development processes are only partially supported, and by only a few methods. Figure 3.3 summarises the coverage provided by some of the methods we examined.

Figure 3.3 Different proprietary methods support different development processes

Method	Type of development process							
	Conventional (linear)	Iterative	Small systems	Accelerated*	Application-package	End-user	Specialist	Emergency
Information Engineering	✓	✓	✓		✓	✓		✓
JSP	✓	✓	✓				✓	✓
Prism	✓		✓		✓	✓		✓
Slim	✓	✓	✓				✓	✓
Yourdon	✓	✓	✓				✓	✓

*Accelerated development typically relies on the use of system-building tools such as fourth-generation languages.

SUPPORT FOR ITERATIVE DEVELOPMENT

Most suppliers of proprietary development methods recognise the need for iterative development. For example:

- James Martin's Information Engineering allows prototyping and structured-decomposition techniques to be used for iterative development, and recognises that it is a possible development process.
- The Yourdon method also recognises a form of iterative development (called radical top-down development), and again provides some supporting techniques.
- The techniques inherent in Jackson System Development and Jackson Structured Programming could also be used for iterative development.

Usually, however, their products do not provide detailed standards and procedures for this type of development process.

SUPPORT FOR SMALL-SYSTEMS DEVELOPMENT

Many proprietary methods can be used during small-systems development, and suppliers have realised the importance of catering for a 'cut-down' version of the life cycle. This avoids burdening a small project with unnecessarily complex and bureaucratic procedures that can be justified only in large-scale projects.

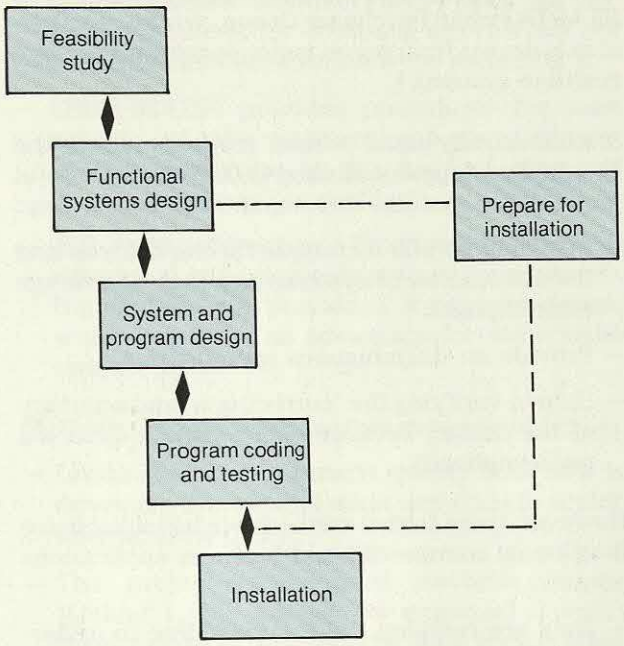
Some methods offer an alternative version specifically for small systems, with its own life cycle and associated standard forms. Hoskyns' Prism, for example, offers the Small-Project Development Method (SPDM). This is designed for projects typically lasting about six elapsed months and involving little new technology or hardware. SPDM is based on the same philosophy as Prism, but it specifies fewer review points because of the reduced risk associated with small projects. The main phases of SPDM are shown in Figure 3.4.

The integrated methods such as Information Engineering and LSDM/SSADM also offer small-systems methods. The latter is called LSDM Fastpath and is a reduced version of LSDM for use with smaller projects with tight timescales.

SUPPORT FOR ACCELERATED DEVELOPMENT

Suppliers recognise the need for an accelerated development process, but they do not provide methods specifically designed for this type of development. Accelerated development typically relies more on the use of system-building tools (such as fourth-generation languages) rather than on methods.

Figure 3.4 Hoskyns' SPDM (Small-Project Development Method) is based on a cut-down version of the life cycle


Feasibility study
Functional systems design
Prepare for installation
System and program design
Program coding and testing
Installation

SUPPORT FOR APPLICATION-PACKAGE DEVELOPMENT

Few proprietary methods incorporate techniques for package selection. In our sample, only BIS Modus, Method 1, Prism, and James Martin's Information Engineering have these techniques.

SUPPORT FOR END-USER DEVELOPMENT

Not surprisingly, few proprietary methods provide guidelines for end-user development. Typically, end users are provided with fourth-generation languages, limited training in development techniques, and access to assistance (often from an information centre) from their development group. Both Modus and Information Engineering provide guidelines on how to manage end-user development, however.

SUPPORT FOR SPECIALIST-SYSTEMS DEVELOPMENT

Most of the well-known proprietary development methods cannot be used to develop specialist (that is, realtime and process-control) systems because the methods are heavily data-oriented and are not activity- or event-based.

However, specific methods have been developed for these specialist systems. These methods are expensive to purchase and use because they have to meet the demanding quality and performance requirements of specialist systems. The 1986 STARTS (Software Tools for Application to Large Real Time Systems) Purchasers' Handbook discusses several methods that can be used for realtime systems (see Figure 3.5) and recommends how some of them could be combined with tools. (This handbook is published in the United Kingdom by the National Computing Centre on behalf of the STARTS Public Purchaser Group, whose membership is drawn from seven major purchasers of large realtime systems.)

Figure 3.5 Realtime methods recommended by the STARTS (Software Tools for Application to Large Real Time Systems) Purchasers' Handbook

Method	*Description*
Core	Core aids the acquisition of information about requirements, preferably (though not necessarily) in conjunction with general-purpose database support tools.
JSD	Jackson System Development aids in systems analysis and design.
JSP	Jackson Structured Programming aids in the design and coding of individual programs based on consideration of the structures of the input and output data.
Mascot	Mascot (Modular Approach to Software Construction, Operation, and Test) is a design method supported by a programming system, supporting the later phases of design and implementation.
Slim	Slim is a project-estimating method.
Price	Price is a project-estimating method.

(Source: 1986 STARTS Purchasers' Handbook)

Mathematically based 'formal' methods such as the Vienna Development Method (VDM) are also useful for realtime systems development because they:

- Provide a formal technique for the analysis and documentation of systems and they encourage clear thought.
- Provide an unambiguous specification.
- Help in verifying the 'correctness' and accuracy of the design, because errors can be detected mathematically.

However, these formal methods are not suitable for developing commercial and business applications because:

- They are complex, take a long time to understand (up to three months for VDM), and are difficult for commercial development staff to learn.
- Users of commercial and business applications are unlikely to understand the documentation that the method requires to be produced.
- It is difficult to measure progress on a project that is being developed with a formal method. Sometimes, it is necessary (for example) to throw away a poor specification and start again.

PROPRIETARY METHODS DO NOT PROVIDE FULL SUPPORT FOR ALL DEVELOPMENT PHASES

None of the proprietary methods we examined fully supports every phase of the conventional development process (Figure 3.6). Apart from project-management methods, no method covers every phase of development; they vary in how much of each phase they address (project-management methods do not cover the how-to-do-it aspects of project management, for example), and they also vary substantially in how they address each phase. Even the so-called integrated methods provide only limited support at the survey and feasibility phase, and they provide no support at the implementation and maintenance phases.

LACK OF SUPPORT FOR ALL PHASES

Although some methods cover most phases of development, no single method covers every phase of the conventional development process. This means that it is not possible to purchase a single method for all aspects of conventional development. It will either be necessary to purchase several methods, each covering some of the phases, or, if one method is purchased, it will be necessary to fill in the gaps with in-house procedures and standards. Typically, those methods that focus on system design and building fail to cover project management and requirements specification, and vice versa. For example:

- The project-management methods (such as BIS Modus, Prism, and Method 1) tend not to extend to the implementation level. This deficiency can partially be overcome by the use of additional tools. For example, if BIS Modus is used in conjunction with BIS IPSE, it provides coverage of most of the phases.
- Although the Yourdon development method provides advice on the use of the project development life cycle, it is designed primarily to provide a set of systems analysis, design, and implementation techniques, and only covers these phases.
- James Martin's Information Engineering covers most phases of the development process, but does not provide detailed project management facilities and techniques. Furthermore, it does not provide facilities to ensure that changes made in one part of the design are consistent with other parts of the design.

In general, the single-phase, analysis and design (multiphase), and integrated methods do not provide project-management facilities. If one or more of these types of method are used in a system development project, the project manager will need to make separate arrangements for project management, perhaps by also using one of the proprietary project-management methods.

There are, of course, project-management methods and tools that have been developed for other types of project but that can be applied to system development projects. For example, PERT (project evaluation and review technique) can be used to identify the critical path of activities within a project, whatever its nature. Other such techniques have either been tailored to suit the needs of system development projects, or have been used as the basis for system development project-management methods.

INCOMPLETE COVERAGE OF INDIVIDUAL PHASES

Methods are often based on some of the techniques necessary to address a phase of the development process but do not cover all of the phase. Moreover, they may define the tasks in a phase, but may not define how the tasks are to be performed; the reverse may also be true. There are several examples of incomplete coverage of phases:

— SADT provides a top-down structured analytical method. This is suitable for systems analysis and design. It can also be used when determining requirements because it provides a graphical documentation facility, which aids communication with users. However, SADT does not provide procedures for obtaining information nor any documentation formats for recording it.

— LSDM/SSADM provides procedures for most aspects of the conventional development process, but stops at the level of 'first-cut' program outlines (or profiles).

— JSP (Jackson Structured Programming) provides structured code that aids system maintenance. But it does not provide a structured design, which would be an advantage for some maintenance jobs.

Methods may also vary in their coverage of phases:

— LSDM/SSADM and Stradis specify both how to develop a system and what activities to undertake at each phase.

— The project-management methods (Prism, Method 1, BIS Modus, for example) typically specify in detail what is to be done at each phase of development. They also provide some details of how each phase is to be performed, although this is not their main focus.

— Yourdon, SDM, and Merise focus mainly on how to do systems analysis, design, and implementation. They do not provide project-management

Figure 3.6 No one type of method completely covers every phase of conventional development

	Type of method									
					Multiphase					
	Management		*Single-phase*		*Analysis and design*		*System-build*		*Integrated*	
Development phase	*How*	*What*	*How*	*What*	*How*	*What*	*How*	*What*	*How*	*What*
Survey and feasibility		▨	▨	▨					▨	▨
Requirements analysis		▨	▨	▨					▨	▨
Systems analysis		▨			▨	▨			▨	▨
Systems design		▨			▨	▨			▨	▨
Programming		▨					▨	▨	▨	▨
Testing		▨					▨	▨	▨	▨
Implementation										
Enhancement and maintenance					▨	▨	▨	▨	▨	▨

▨ Indicates the extent to which each type of method covers each development phase.

standards for what must be done in terms of forms to fill in, lists of deliverables, and so on.

Although proprietary methods at present address neither every aspect of conventional system development, nor different development processes, most of the methods are continually being enhanced to provide greater coverage. For example, new versions of Stradis and LSDM/SSADM are released frequently as these products are continually upgraded and enhanced. The scope of these methods has now been extended to provide project-management and development tools. Ultimately, their suppliers plan to extend their scope into the implementation phase.

DEVELOPMENT METHODS NEED TO BE SUPPORTED BY APPROPRIATE TOOLS

In Chapter 2 (on page 7) we explained that, to be used effectively, development methods need to be supported by appropriate tools that will automate the activities specified by the method. Unfortunately, many proprietary methods suppliers do not provide the necessary tools with their products as a matter of course. Most of the organisations that reported initial dissatisfaction with their use of a proprietary method said that they lacked the tools needed to support the method.

In one company, after about nine months of using a particular method, the analysts expressed extreme dissatisfaction, and the project managers began to use it selectively or not at all. Further investigation showed that the highly bureaucratic and paper-driven method was not supported by tools in this company. The analysts found it tedious and time-consuming to use the method properly, and development productivity was reduced. The method was thus discredited, and they reverted to the original in-house development method.

We also found that some system development managers are not completely familiar with the facilities offered by the various types of tool, and are not fully aware of the need to automate system development methods. They are sometimes unclear about how a specific tool could fit into their development process. Managers are also concerned about investing in development tools because, in this rapidly changing area, tools acquired now could swiftly be made obsolete by new products. Managers therefore need also to be aware of the benefits to be derived from using appropriate tools with their chosen method.

THE NEED FOR PROPER SUPPORT FROM SYSTEM DEVELOPMENT TOOLS

Some of the most valuable techniques provided by system development methods are impossible to use without suitable development tools. The best example of this is prototyping for which high-productivity advanced system-building tools are essential. (This point was explained in Foundation Report 47 — The Effective Use of System Building Tools.) Other examples include estimating packages that draw on large historical databases, and analyst workbenches that permit online development of data models and that need to be underpinned by databases.

Without appropriate tools, some methods will not be used, or will be used reluctantly by developers, because, without tools, they are very time-consuming and tedious to use. Drawing data-flow diagrams, entity diagrams, functional-decomposition charts, and so forth by hand is an extremely laborious process because many diagrams are required and most of them have to be changed several times.

If the structured methods are not accompanied by an appropriate software tool, systems analysts and designers often quickly lose sight of the advantages of the methods and see the process as a tedious bureaucratic overhead. In these cases, whether the method continues to be used or not depends on how much freedom the developers have to determine their own working practices.

In one organisation, a large government department in the United Kingdom, developers were told to continue using a particular method, despite their objections and dissatisfaction with it. Eventually, the department provided them with appropriate tools. Productivity was low during the interim period, however.

On the other hand, in some commercial organisations, only token attention is given to the use of a method. For example, in a major multinational engineering company there was an officially recognised system development method, but development staff continued to analyse applications in the 'traditional' way. At the end of the analysis they produced documentation to the standard imposed by the method, but they saw this as a documentation task, not an analysis activity. They were thus effectively not using the method at all, but were merely adding to their documentation workload. The situation changed when a basic analyst/designer workbench supported by a data dictionary was made available to the analysts, and development productivity improved considerably. In addition, development costs were reduced and project schedules were adhered to.

SEVERAL TYPES OF TOOL ARE AVAILABLE

Our research showed that, because of the confusion that exists in some people's minds about the distinction between development methods and tools, there is a belief that a single tool can be used throughout the whole development process. Unfortunately, this is not true. Different tools are needed to support different development methods, and, as we have already shown, no method covers all phases and all types of development process. We have identified five basic types of development tool:

- Project management tools, such as timesheet analysers, project-estimating packages, project-control and planning packages, and productivity-measurement tools. Examples of proprietary tools include Artemis, Prompt, and Project Manager Workbench (PMW).
- Analyst/designer workbenches, which support the systems analysis and design phases of development. Examples of proprietary products include Core Analyst, Automate, Datamate, Yourdon's Workbench, Index Technology's Excelerator, Prokit Analyst, Design 1, Emeraude, Alcide, and Speedbuilder.
- Programmer tools (such as Delta and PDF), which support the programming phase of development.
- Advanced system-building tools such as Linc, Mapper, Focus, Natural, and Powerhouse. This type of tool was discussed in detail in Foundation Report 47 — The Effective Use of System Building Tools.
- Integrated project support environments (IPSEs), claimed by their suppliers to support the whole of the development process. Proprietary examples include IEF, BIS IPSE, ISTAR, and Philips' Maestro.

The tools we reviewed during the research are listed in Figure 3.7. The range of facilities offered by development tools, together with examples of products providing appropriate facilities, are shown in Figure 3.8.

NO CURRENTLY AVAILABLE TOOL SUPPORTS ALL PHASES OF THE DEVELOPMENT PROCESS

Despite the extensive publicity given recently to the emergence of a new family of tools, the integrated project support environments (IPSEs), none of the currently available development tools provides complete support for all phases of the system development process. Furthermore, the facilities provided by tools are evolving rapidly and

Figure 3.7 Development tools reviewed during the research

Tool	*Supplier*
Alcide	METSI, Delta Software Ltd
Artemis	Metier Management Systems Ltd
Automate	LBMS (Learmonth & Burchett Management Systems)
BIS IPSE	BIS
Core Analyst	System Designers plc
Datamate	LBMS (Learmonth & Burchett Management Systems)
Delta	Delta Software Tools Ltd
Design 1	Arthur Andersen & Co
Emeraude	Emeraude Gie (Paris)
Excelerator	Excelerator Software Products Ltd, Index Technology Corporation
IEF (Information Engineering Facility)	James Martin Associates
ISTAR	Imperial Software Technology Ltd
Maestro	Philips
PDF (Progress Development Facility)	Michael Jackson Systems Ltd
PMW (Project Manager Workbench)	Hoskyns Group Ltd
Prokit*ANALYST	McDonnell Douglas
Prompt	LBMS (Learmonth & Burchett Management Systems)
Speedbuilder	Michael Jackson Systems Ltd
Yourdon Workbench	Yourdon Europe

Figure 3.8 Range of facilities provided by system development tools

Type of facility	*Tools providing appropriate facilities*
Administrative	Maestro provides facilities for electronic mail, standard forms, diary management, text handling, and version control of documentation.
Multi-user	Maestro, ISTAR, and BIS IPSE all provide a multi-user environment. (Note: Delta, Speedbuilder, and PDF naturally exist in a multi-user environment).
Project management	Artemis, Prompt, and PMW all provide extensive project management capabilities: – Artemis is most suitable for large projects. – PMW provides a high level of functionality but is PC-based (as is Prompt).
Requirements analysis	Core Analyst supports Core and is a sophisticated tool for requirements analysis.
System analysis/ design	Many tools provide system analysis/ design facilities. As yet, no one tool appears to be superior to the others, although Excelerator probably has the largest worldwide market share.
Implementation	Implementation tools vary widely, but IEF is the first tool to offer automatic code generation.

none of the tools incorporates all the latest best practices.

To support an organisation's development methods, the tools chosen must provide support for the whole of the conventional development process and for any other development processes being used. How well different tools provide this support, and what the state of the art of tool development is, are discussed below.

Proprietary tools support most development processes but only to a limited extent
Most proprietary tools support most of the development processes (except end-user and realtime) to a limited extent. The extent of their support is dependent on the nature of the tool. For example, project-management and administration tools can be used for any project, and analyst/designer workbenches and programmer tools can be used irrespective of whether a conventional, iterative, or small-systems development process is being used.

Many proprietary tools do not support the end-user development process because they have been designed for use by professional development staff, and require knowledge about, and understanding of, the management and development techniques associated with the method being used. End users would not usually have this level of expertise. Some tools are suitable for end users, however, particularly system-building tools like Focus.

Specific realtime development tools like the ISTAR IPSE product are available to support specialist development processes, but commercial tools tend not to support specialised realtime methods.

Proprietary tools do not support the complete conventional development process
Figure 3.9 shows the extent to which the different types of proprietary development tool support the various phases of the conventional development process. Different tools typically support only a specific aspect of system development such as project management, analysis/design, or implementation. Even the integrated tools (IPSEs) do not support the whole of the development process.

THE STATE OF THE ART IN TOOL DEVELOPMENT

System development methods are evolving rapidly. This means that any tool or tools chosen today should support the best latest practices. However, no proprietary tool currently provides all the facilities that might be required. Nevertheless, there are three facilities that we consider to be the most important, and development managers should ensure that they are available when choosing proprietary tools:

— A multi-user facility that allows developers to enjoy the benefits of administrative and project-control facilities, and enables them to share information about development projects.

Figure 3.9 No one type of tool completely supports every phase of conventional development

	Type of tool									
	Project management		*Analyst/designer workbenches*		*Programming*		*Advanced system-building tools*		*IPSEs*	
Development phase	*How*	*What*	*How*	*What*	*How*	*What*	*How*	*What*	*How*	*What*
Survey and feasibility		▨								
Requirements analysis		▨	▨				▨			
Systems analysis		▨	▨				▨		▨	▨
Systems design		▨	▨		▨		▨		▨	▨
Programming		▨			▨		▨		▨	▨
Testing		▨			▨		▨		▨	▨
Implementation		▨			▨		▨			
Enhancement and maintenance			▨		▨		▨		▨	▨

▨ Indicates the extent to which each type of tool covers each development phase.

– Graphical facilities that can be used to represent models of the system during the analysis and design phases, supported by databases, automatic consistency checks, and rules for ensuring that the method is followed.

– Ability to generate code automatically from detailed design documentation.

THE BENEFITS GAINED FROM SUPPORTING METHODS WITH TOOLS

Using tools to automate the activities within a method also provides secondary benefits during system development. Different types of tools provide different benefits:

– Project-management tools and IPSEs allow the project to be planned and controlled more effectively.

– Analyst/designer workbenches, programmer tools, and some IPSEs help to improve the productivity of development staff.

– IPSEs and some design tools help to improve communication between staff working on a project.

Project-management tools improve project planning and control

A project-management facility allows the project manager to plan and control a development project more effectively because:

– Project estimating is improved by tools that require a disciplined approach to the collection of historical and current project data. These tools also enable estimates to be updated rapidly in the light of increased knowledge and changing requirements, and they make it possible to produce different estimates based on different parameters (system size and type, difficulty of project, manpower level, and scheduling constraints).

– Project planning and reporting is improved by tools that allow resources to be assigned (and easily re-assigned); 'what-if' scenarios to be incorporated; a range of different graphical reports to be produced quickly both for systems and user management and for the development staff involved in the project.

– Project control is improved by tools that quickly highlight any deviations from the original plan.

Tools improve development productivity

The introduction of a proprietary development method will not, by itself, improve the productivity of development staff. Productivity improvements will only occur if tools that support the method are also installed. Development tools can improve development productivity in the following ways:

— Advanced system-building tools provide productivity advantages during the implementation phase. This is the most significant productivity benefit offered by tools.

— Analyst/designer workbenches (or word processing and graphics packages) speed up the production of the analysis and design documentation.

— In large systems, it is difficult to cross-check information manually (for example data names) within each phase and between phases, and it is easy to make mistakes. Some tools (IPSEs and some analyst/designer workbenches) do this cross-checking automatically and save the analyst/designer a substantial amount of clerical effort.

— Tools such as the IPSEs also allow code to be generated automatically from the detailed design documentation.

The overall impact of analyst workbenches is difficult to assess because they have been in use for only a limited period and are evolving rapidly. We suspect, however, that their impact will be limited mainly to the design phase. Their impact on total development productivity will become substantial only when the workbench output can be used directly by system-building tools.

IPSEs and some design tools improve communication

Using tools to support the method can also improve communication among the staff working on a project. For example:

— The office automation facilities provided by multi-user workbenches speed up communication between members of the development team. This is particularly important on large development projects and those where members of the team are located at more than one site. If the system's users have access to the same office automation facilities, then communication with them may be improved as well.

— Good analysts are not necessarily good draughtsmen and the effectiveness of using the graphical techniques in systems documentation can be reduced if analysts produce messy charts and diagrams. Many development tools provide graphical facilities that help the analysts to prepare neat and orderly diagrams.

— Some tools (for example Maestro) provide common sets of documentation (standards, formats, and so on) that are available immediately for electronic distribution to all members of the development team.

SUMMARY

In this chapter we have shown that a proprietary method by itself will provide an incomplete solution to an organisation's system development problems. A single method will not be suitable for all types of development process, and, even where it can be used, it will not cover all of the development phases. Moreover, development tools have to be used in conjunction with a method if the most effective use is to be made of the method.

The proprietary tools available today are also limited in their application because they too cannot be used with all types of development process and they do not cover all phases of development.

To make the most effective use of proprietary development methods, an organisation must therefore choose methods and their supporting tools and integrate them with its overall development processes. We offer advice on how to do this in the next chapter.

Chapter 4
Selecting appropriate methods and tools

There is now a wide range of proprietary development methods and tools available in the marketplace. The choice of a method and the tools to support it depends on whether they support the development processes being used by the organisation, and on the techniques on which the method is based. It also depends on whether the method and tools can be used with the hardware environments in which the systems will be developed, and on the level of support provided by the suppliers.

Implementing a new method can be a major undertaking. Development staff (and in some cases users as well) need to be educated and trained in the new method. Hence, the training and support provided by the supplier is an important consideration when choosing a proprietary method. Some suppliers offer a turnkey service, where they guarantee to train all the development staff in the use of the method, and help with its introduction.

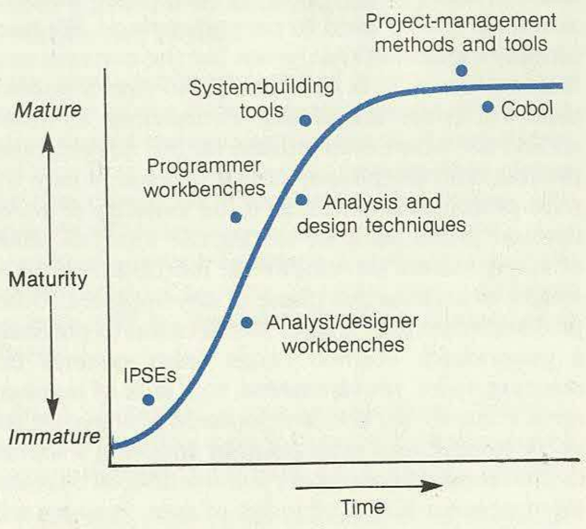
System development managers find it difficult to evaluate the relative merits of different proprietary methods and tools because the suppliers tend to overemphasise the benefits of their own particular product relative to others. Some bring an almost religious fervour to the sale of their method or tool, promoting it as the solution to all the developer's needs. Thus, a method covering only one or two phases may be presented as a comprehensive method; a collection of associated analysis and design tools can be promoted as a totally integrated method; or a good data processing system development method can be promoted as being suitable for applications of other kinds.

These exaggerated claims are a symptom of the immaturity both of the products available and of the market. Most methods are continually being enhanced to embrace the latest techniques and tools emerging from the research on systems development. For example, a new systems-analysis and design concept is that of the 'entity-life-history'. When we carried out the research for this report, none of the proprietary methods we examined were based on the entity-life-history concept, although such methods will undoubtedly appear in due course. Similarly, the supporting tools are undergoing rapid development.

The result is that different techniques, methods, and tools are at different stages of maturity. Figure 4.1 shows the relative maturity of some techniques, methods, and tools. Most suppliers of methods and tools provide new versions of their products regularly, as they try to keep abreast of the technical developments. Indeed, some update their product every three to six months, which means that any technical review of methods and tools quickly becomes out of date. Hence, any choice of method or tool must take account of not only what is available now, but the likely developments of the product over the next few years.

Before setting out to select proprietary methods and the tools with which to support them, an organisation must first be clear about the different types of development process that will be used. Some methods and tools cannot be used with particular types of process. And no method or tool even covers the complete conventional (linear) process.

Figure 4.1 Different development techniques, methods, and tools are at different stages in the life cycle



IDENTIFY THE DEVELOPMENT PROCESSES TO BE USED

In addition to the conventional development process, most organisations will also develop systems using one or more of the other types of development process identified in Chapter 2. Some types of application (decision-support systems and those developed by end users) are best developed by a process other than the conventional linear process.

The types of development process in use, and the phases within each process, will be determined largely by the overall style and culture of the organisation. A bureaucratic organisation will tend to develop systems differently from an organisation that has a much more laissez-faire approach to the way it conducts its business. And one-off applications designed to provide specific information to satisfy a particular requirement will be developed in a way different from applications that will be used regularly over many years.

It is not possible to provide general advice about the number and types of development process that should be adopted, other than to say that the number of processes formally recognised by an organisation should be kept as small as possible, consistent with the total applications needs. We believe that most organisations will not be able to standardise on the conventional process alone. Yet adopting more than one process has evident disadvantages. Additional training and implementation costs will be incurred for every extra process, for example.

DECIDE WHERE PROPRIETARY METHODS CAN BEST BE USED

Having determined the development processes that will be used, the next step is to decide where in each process proprietary development methods and tools can be used to best advantage. We have already explained that, even for the conventional linear process, it is not possible to find a method that will cover all phases. Foundation members should therefore concentrate on the development process, and the phases of that process, where the most problems arise. Thus, if the majority of development problems arise during the analysis phase of conventional development, methods should be sought to address this phase of development. Other problems that may trigger the decision to purchase a proprietary method range from systems not meeting users' requirements, to a lack of management control over the development process. On the other hand, there is no point in adopting a sophisticated method to support a conventional development process if the majority of new systems will be based on packages.

Decisions about methods to support other development processes and phases can then be postponed, awaiting the improvement in methods and tools that will occur in the next twelve to eighteen months. These improvements will make it more likely that a method can be found to cover several development processes and more phases of the processes. In particular, some of the tools, like analyst/designer workbenches and IPSEs will improve considerably over the next year or so. These types of tool have only recently become available as commercial products, and they are being continually enhanced by their suppliers. Although tools such as IPSEs are now being used increasingly, many organisations prefer others to be the pioneers in using this novel technology.

The experience of other organisations is valuable when considering whether to purchase a proprietary method. Some of the reasons why organisations that participated in the research introduced a new method are briefly described below:

- A multinational oil company introduced a management method in order to improve the quality of its systems documentation. The maintenance of old systems was a major problem, mainly because of a lack of documentation, and maintenance costs were rising as a proportion of the total expenditure on systems. The company introduced a bureaucratic forms-driven method to overcome this problem.
- A large retail company introduced Method 1 as part of a major reorganisation of the systems division. This reorganisation was prompted by extreme user dissatisfaction with the quality of systems being produced.
- A major bank introduced LSDM to reduce the large number of bugs found during the implementation of systems.

These examples illustrate that organisations choose proprietary methods to help them to overcome specific development problems. However, the use of methods is not the only way to solve development problems. A problem with systems quality, for example, could also be addressed by:

- Using prototyping for the requirements-definition phase.
- Improving staff training.
- Recruiting better staff.

CHOOSE AN APPROPRIATE METHOD

After identifying the development processes, and the phases of those processes, for which methods are required, the next step is to choose the most appropriate methods. Most organisations will now evaluate the alternative proprietary packages that are in the marketplace, selecting the products that best match their requirements. A few organisations, however, have attempted to construct their own in-house method and the supporting tools. We would not recommend this approach because it is expensive, time-consuming, and very difficult. It also requires considerable in-house expertise.

In addition, an organisation that constructs its own method and tools may not be able to benefit from technical advances made in proprietary methods and tools because the investment required to update its own procedures is prohibitive.

In the early 1980s, a large American-based international engineering company attempted to develop an in-house process for system development, incorporating both methods and tools. After a considerable investment of staff time (25 man-years) and consultancy and training costs, the attempt was abandoned and proprietary methods were chosen and introduced. This company's experience illustrates the risks of attempting to develop methods and tools from scratch. It is technically complex, requires high levels of investment, diverts staff from application development, and success is not guaranteed.

We would expect the cost of a proprietary method to be less than that of developing an in-house method because the development costs can be shared amongst the purchasers. For most organisations, the only sensible course of action therefore is to choose a proprietary method. There are two main options:

— Choose an integrated method and the tools to support it.

— Choose one or more proprietary methods to address specific phases of the development process.

Despite the claims of the suppliers, integrated methods do not yet cover all development phases. We believe that the best approach is to configure your own development process, making use of proprietary methods wherever it is appropriate to do so.

INTEGRATED DEVELOPMENT METHODS ARE RISKY

A few suppliers claim their 'integrated' proprietary methods and tools cover the whole of the conventional development process. Two examples are BIS (with its Modus and IPSE Products) and James Martin Associates (with Information Engineering and IEF — Information Engineering Facility). As yet, there is only a limited amount of experience with either of these methods. For example, in March 1987 BIS IPSE was in use at only 20 sites, and IEF is still being beta-site-tested.

Although there is not yet any hard evidence to show how successful these integrated methods are, initial reports are encouraging. For example, Pearl Assurance, a leading UK life assurance company, is very satisfied with its use of BIS IPSE. Pearl believes that the benefits it has gained in practice are even greater than were originally planned.

What is clear, however, is that adopting an integrated method will require extensive support from the supplier in terms of both consultancy help and staff training. To a great extent, the success achieved will depend on the organisation's willingness to be flexible and change its current development process to the one prescribed by the method. Changing the integrated method and tools to meet its own requirements may not only be costly, but it will also introduce a risk that the method will not be used effectively. Furthermore, if the method and tools are changed, the organisation may not be able to take advantage of any new versions released by the supplier.

There are other disadvantages of buying an integrated method. As Chapter 3 explained, no system development method — even the integrated methods — at present covers all of the phases of the development process. Also, adopting an integrated method and tools is technically risky. These products are at a very early stage of development, and later products coming onto the market are likely to be technically superior.

Unless an integrated method can be used to solve a particular development problem, we advise Foundation members not to adopt integrated methods in the short term, but wait until these products are proven.

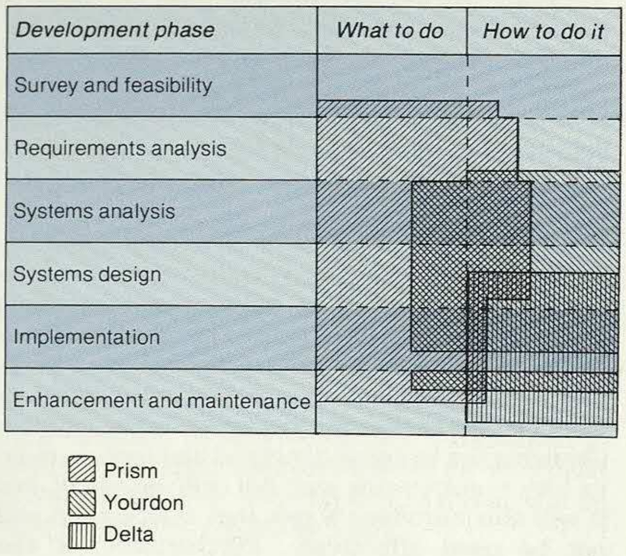
SELECT METHODS TO ADDRESS SPECIFIC DEVELOPMENT PHASES

In the absence of an integrated method that covers the whole of the development process, an organisation should choose a combination of proprietary methods that address specific development phases and that complement each other. Sometimes, two methods will be required for the same phase — one to address what-to-do issues, the other to address the how-to-do-it aspects. In other cases, different methods will be used for different development phases.

Figure 4.2 (overleaf) shows how the Prism, Yourdon, and Delta products were integrated by one company to cover most of the system development process:

— Prism addresses project management, the initial survey and feasibility phase, the requirements-

Figure 4.2 Example of complementary methods providing (almost) complete coverage of the conventional development process



analysis phase (detailing what has to be done and how to do it), and provides guidelines about what should be done during the system analysis and design phases.

— Yourdon provides structured techniques for the system analysis, design, implementation, and maintenance phases.

— Delta is a tool that supports structured programming and maintenance.

There are areas of overlap between the components, particularly in the system analysis and design phases, which are covered both by Prism and Yourdon. In this case:

— Prism would be used to determine what should be done during the system analysis and design phases because it provides more detail in these areas than Yourdon.

— Yourdon would be used to determine how systems analysis and design should be carried out because it provides more detailed procedures.

Occasionally, different methods are used in conjunction with one another, even though they provide incomplete coverage of the system development process. An example is the use of MCP in conjunction with Merise. MCP provides basic project-management techniques. It identifies the phases of the life cycle but does not identify details of the tasks and deliverables from each phase. Merise provides system analysis and design techniques. However, when they are used together, they do not cover the whole of the development process. Figure 4.3 shows the gaps that are left.

In such cases, the system development department will need to use in-house standards and procedures to fill in the gaps left by the proprietary methods. The danger is that individual development staff will use their own methods, based on their own experience, to fill in the gaps. If different developers working on the same project use different in-house methods, the ensuing confusion may well be worse than the problems that the introduction of proprietary methods was meant to solve.

Some proprietary methods should not be used together because they cover the same phases of the system development process and they address the same aspects of the phase, both defining either what is to be done or how will it be done. For example, both Prism and BIS Modus are project-management methods. They both provide guidelines on what is to be done at each phase of development, from the survey and feasibility phase through to detailed system design (see Figure 4.4). They both state how project management is to be carried out and provide techniques for the survey, feasibility, and requirements-analysis phases. Neither of them provides detailed techniques for the system analysis and design phases, although both recommend the use of structured techniques. BIS Modus, however, can be used with BIS IPSE, which has an analyst/designer workbench supporting structured techniques.

The advantage of mixing and matching different proprietary methods and tools is that a much closer fit to the needs of the development group can be achieved. But the choice of methods needs to be made carefully to avoid the possible problems described above. In addition, it is advisable to carry out small trials lasting no more than about four or six weeks before deciding that a particular method

Figure 4.3 Example of complementary methods providing incomplete coverage of the conventional development process

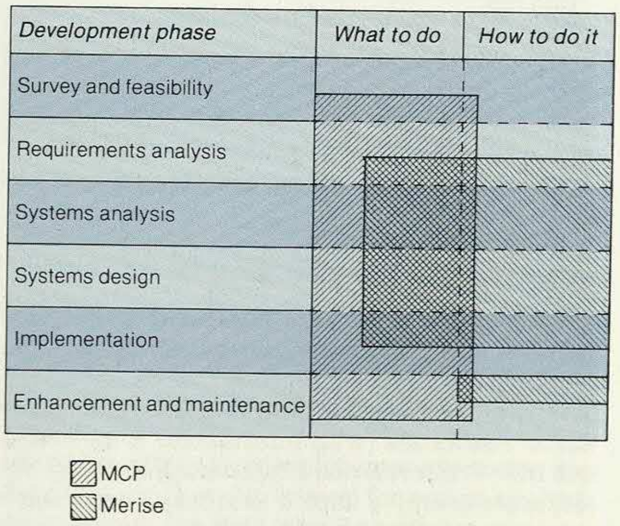
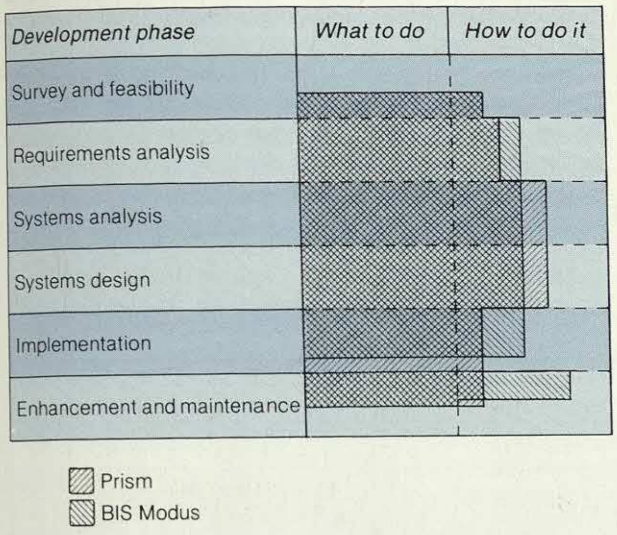


Figure 4.4 Example of overlapping methods



is appropriate. These trials should be carried out by experienced system development staff and should be monitored closely by the systems development manager. Most method suppliers welcome this type of experimentation period, and will often provide extensive support for it. The results of the experiments should be reviewed by management as quickly as possible after the end of the experimentation period.

CRITERIA FOR SELECTING A METHOD

Apart from ensuring that the technical facilities provided by a development method are adequate, there are two main criteria for selecting a proprietary method:

— The level of support provided by the supplier.

— The 'fit' with the corporate culture.

ENSURE THAT METHODS ARE SUPPORTED BY SUPPLIERS

The consultancy and training services offered by the method supplier were quoted by many users as the most critical factor in selecting a method. The commercial stability and technological track record of the supplier is also of concern.

Most suppliers of methods provide training courses in the use of their method. These are typically available either as public courses, or as in-house courses, if required. (Indeed, for many suppliers, training provides their major source of revenue.)

When choosing a method, it is important to ensure that frequent courses in its use are provided because there will be a recurring need to train staff. There are several major characteristics to look for in these courses:

— There should be a variety of courses available for management, users, and the different types of development staff who will use the method (analysts, designers, programmers, and so on).

— The training programme should be updated in line with improvements to the method.

— Clear and extensive training material should be provided.

— Where appropriate, the use of automated tools to support the method should be demonstrated and taught.

— The training staff should have experience of and expertise in using the method.

The purchaser of a proprietary method should recognise that training costs represent the largest proportion of the total cost of installing a method, and that these costs will continue because new staff will need to be trained in the method.

Many organisations recognise that the consultancy services of the supplier are required to ensure that a method is properly used, particularly when it is first implemented. Some suppliers (BIS, for example) offer a complete turnkey service for the installation of their method. Sometimes, a consultant from the supplier works with the development group on a pilot application, and is therefore available to give advice and help.

A checklist of what to expect from method suppliers and some further pointers to gaining the best value from a supplier are shown in Figure 4.5 overleaf.

The supplier should also have local staff available who have experience in the use of the method, and who can provide the required level of consultancy, advice, and support. A concern expressed by European users of one method was that all the experts were based in the United States. This situation should be avoided when choosing a method.

The commercial stability and technological track record of the supplier are also of concern. There are two main reasons for this:

— Training and consultancy are needed from the supplier on a continuing basis. The supplier must be large enough and financially secure enough to ensure that it can continue in business for the foreseeable future. If the supplier ceases to trade, the substantial investment made by an organisation in implementing the method may have to be written off.

— Ideally, the method should be 'mature', having been implemented successfully in several other organisations. Any new or immature method should be selected only if it offers substantial technical advantages. It is also vital that the supplier be prepared to continue to develop the method to incorporate new concepts and techniques.

CHOOSE METHODS THAT 'FIT' WITH THE CORPORATE CULTURE

The organisation's corporate culture is an important factor in selecting a proprietary development method. System development methods formalise the procedures for developing systems. To be successful, the procedures must either complement or replicate the working practices commonly used in the organisation. If the selected method requires the organisation to depart significantly from its customary practices, then the method is unlikely to be accepted. The main organisational characteristics that influence the choice of a system development method are the extent to which the organisation is structured as a hierarchy, and how paper-oriented it is. For example:

— Large, bureaucratic organisations tend to choose, and have success with, those methods

Figure 4.5 What to expect from the method suppliers

Product feature	*Description*	*Pricing policy*	*Other comments*
Documentation	Manuals describing: — In general, the method and those phases of the development cycle it covers. — Step by step, for each major activity, what needs to be done and how to do it. — For each milestone, the deliverables (typically systems documentation), with worked examples.	This is the basic product, and it is normally sold at a fixed price, with, perhaps, some annual charge to cover updates.	Be aware that this documentation, which is normally voluminous, is on the suppliers' word processor. Any amendments will, therefore, be made physically by the supplier (irrespective of who prepared the amendment) and the client will incur word processing, printing, and distribution costs (often at alarmingly uncompetitive rates). It is possible to negotiate for the source documents to be converted to the client's word processor — with rights to amend it. This is also generally expensive.
Tailoring	Amending the method (especially the documentation) to fit the client's environment more closely. This might involve including any equipment or software restrictions, for example.	Time and materials at consultancy rates — apart perhaps for a limited number of days of free support included with the basic product. Word processing, printing, and distribution charges may also be applicable.	Tailoring generally must occur before implementation. As a consequence, the client is likely to have too little experience of the product to tailor it and will be dependent on the supplier. This is where the suppliers make their profit. Try to get a fixed price on this potentially uncontrollable expense.
Piloting	Assisting the project team to pilot the method through the first application.	Time and materials at consultancy rates.	Specific terms of reference are essential. Again, try for a fixed price. Be aware that the success of the pilot may be attributed to the supplier's facilitator — enthusiasm for the project may wane once he is off-site.
Training	Preparation and presentation of courses on the method for systems staff and users.	Fixed price per course — depending upon duration and location.	Another potentially large expense. Try to have courses on your own site; secure ownership of the material and become self-sufficient as soon as possible.
Ongoing support	Assisting systems staff after implementation.	Time and materials at consultancy rates.	Do not remain dependent on the supplier. Establish your own, in-house support unit.
Software	Providing software tools to support the method	Definitely optional. May be bought from a different supplier. In any case, pricing may be either once-off with an annual maintenance contract, or monthly rental. Any documentation (up to a specified limit) may be included in the software price — but not other requirements (like piloting, training etc).	Be aware that much of this software runs on microcomputers and the licence may be per machine. To make the software available to all users in the systems department can be very expensive.

that are heavily biased towards project management and control and are forms-driven. A good example of this is the United Kingdom Government's use of SSADM.

- Alternatively, results-oriented companies in fast-moving environments (for example the retail sector) have greater success in implementing methods that place relatively little emphasis on project management, relying instead on skilled staff using advanced techniques and tools (like using a fourth-generation language for prototyping) to develop systems.

Some systems departments have adopted a bureaucratic style as a defensive measure to counter user criticisms. Sometimes the line managers would like a fast response to their needs, but have abandoned hope, and interest, through years of poor experience. The introduction of a new development method may make it possible for the systems department to adopt a more entrepreneurial attitude, thereby enabling a better service to be provided and good relations with the line managers to be re-established. Thus, new methods may themselves affect the corporate culture, at least in so far as it concerns the systems department.

CHOOSE TOOLS TO SUPPORT THE METHODS

The final stage in selecting the appropriate system development methods is to choose the tools that will provide automated support for the methods. In some cases, the tools are provided along with the method. However, if a mixture of methods has been chosen, it is likely that the system development manager will need a selection of tools to provide the support for the methods.

Some method suppliers recommend tools that are suitable for supporting their methods. For example:

- Speedbuilder and PDF are recommended for use with Jackson System Development and Jackson Structured Programming.
- Prompt, Automate, and Datamate are recommended for use with LSDM/SSADM.
- Design 1 is recommended for use with Method 1.

The advantages of choosing supplier-approved tools are that the tools are designed to integrate with each other and with the method, and any changes made to one of them are incorporated in the others. Furthermore, the supplier typically supports both the tool and the method. However, it is possible to replace selected tools with others that undertake the same tasks but that offer different and even improved facilities (for example, replacing Automate with Excelerator, or Prompt with PMW). Some technical expertise is required for this to be done successfully, however.

The major disadvantage of choosing a range of tools is that they will seldom offer the same advantages as a true IPSE (automatic generation of code, for example) and they often do not have automatic interfaces between then. This means that substantial nonproductive clerical coding effort may well still be required.

The tools chosen should have a range of technical facilities appropriate to the phase (or phases) of the development process that the tools are to support. The range of technical facilities for development tools was listed in Figure 3.8 on page 15.

Once the development processes that will be used have been decided on, and the development methods that will be used and the tools that will support the methods have been chosen, the next stage is to manage the introduction of the new methods. This is a critical task, requiring substantial effort. We turn our attention to this topic in the next chapter.

Chapter 5

Managing the implementation of methods

Successfully introducing proprietary methods, perhaps for several different development processes, is a substantial task and is likely to be both expensive and time-consuming. It requires effort and commitment, not only from systems development management and staff, but also from user management and staff. The key to success in implementing a new method is therefore to manage its introduction actively. We provide guidelines on how to do this in this chapter. There are four critical steps that have to be taken:

— Obtain senior user management consent.

— Assess the impact of the methods on the organisation of the systems department.

— Gain experience by using the new methods for pilot applications.

— Implement the new methods and tools as a distinct project.

Each of these is discussed in more detail below.

OBTAIN SENIOR MANAGEMENT CONSENT

Before introducing the new method or methods it is vital to gain the consent of senior user management. This is necessary because:

— Implementing the method will require substantial effort from development management and staff, and in the short term this will divert effort from developing new applications.

— When the method is implemented, the basis of the relationship between user management and staff and development staff is likely to change, with users being much more closely involved in the development process. User management needs to be aware of the organisational and personnel implications of this change.

— Investing in a new method is likely to be expensive. Not only is there the cost of purchasing the method and tools themselves, there is also the cost of the training and consultancy required, the cost of piloting the method, and the 'lost-opportunity' time of developers and users as they learn to use the method.

CONSIDER THE ORGANISATION OF THE SYSTEMS DEPARTMENT

Before implementing a method, the current organisation of the systems development department must be considered in order to assess the likely impact of the method, and to prepare for any organisational changes that may be required.

Most development groups are organised in one of three ways:

— By business function (specialist groups for accounting, manufacturing, and so on).

— By project team, to develop specific applications.

— By job specialisation (business analysts, systems analysts, systems designers, and so forth).

Many organisations have a mixture of these structures in their system development department. When choosing a method, it is important to consider the current organisation and the impact of the method on it in order to avoid an inappropriate choice. For example:

— A development group based wholly on job specialisations would find it difficult to carry out iterative development because the system-builder or analyst/programmer skills are not readily available.

— A systems department based on small project teams of development staff who have expertise in developing small systems quickly is likely to find that large project-management-based methods are inappropriate.

The introduction of a new method is also likely to change the roles played by the development staff, with more emphasis being placed on project-management and communication skills and less on technical skills like coding. Inevitably, the roles of the analyst, designer, and programmer will begin to converge. This will be a direct consequence of the increasing automation of the development process, the increased use of prototyping, and the use of advanced system-building tools in the implementation phase.

GAIN EXPERIENCE IN USING THE NEW METHODS

The introduction of a new method will change the way in which all system development staff carry out their work. Before implementing a method throughout the development department, it is necessary to gain some real experience of using the method and to create a nucleus of staff who are expert in using it. Such experience and expertise cannot be gained quickly, and carefully planned pilot projects need to be carried out. Whilst these pilots are carried out, all other systems will be developed using whatever methods are currently in use. Depending on the development processes that the new method is to be used with, any of the following types of pilot project may be required:

— A small-systems development project lasting about six months.

— An iterative development project, with the development being evaluated after about nine months.

— A conventional development project lasting between 12 and 18 months.

It is important to select the pilot project teams carefully, with the team members representing the development group as a whole. Thus they should have an average level of expertise and experience; they should not be the best and brightest development staff. At the same time, the team members should not be opposed to the introduction of the new method. Eventually, some of the pilot team members should be able to help with the wider implementation of the method.

For similar reasons, the applications chosen for the pilot projects should be typical of the systems department's development workload. However, they should not involve a high business risk, because additional risks will inevitably be introduced by using the new method for the first time.

Before the pilot projects begin, the team members should be thoroughly trained in the use of the method and its supporting tools. Supplier support must be available throughout the pilot projects, ranging from full-time consultancy assistance to offering specific advice and help when necessary.

At the end of each pilot project, success should be evaluated by:

— Comparing each project with similar projects carried out using the 'old' methods, in terms of productivity, user satisfaction, cost, timeliness, and number of errors after implementation.

— Obtaining the views of the pilot development teams on both the advantages and disadvantages of the method and tools compared with the current methods, and on what particular problems the new method posed during the pilot projects.

— Seeking the views of the users on their perception of the development process, and on their opinion of the resultant application systems.

Once the pilot applications have been completed and evaluated, and the lessons have been learned, the method can then be implemented throughout the system development department.

IMPLEMENT A NEW METHOD AS A DISTINCT PROJECT

The implementation of a new method throughout the systems development department should be viewed as a project in its own right. A project team should be formed to manage the implementation, and a formal implementation plan drawn up. The project team should be small, with around three or four members. The members of the team can be drawn either from the initial team that evaluated the methods and/or from the pilot teams. It is essential, however, that the manager is an effective project manager and that the team members are familiar with the technicalities of the method and tools.

The responsibilities of the implementation project team include:

— Developing the implementation plan.

— Liaising with the method and tool suppliers.

— Creating guidelines for using the methods.

— Assisting in the education and training programmes.

— Providing continuing support (together with the suppliers) to the development staff who are using the method and tools.

— Appraising the success of the implementation project, and modifying and extending the guidelines if necessary.

The implementation plan consists of three separate but related components:

— An education and training plan for development management and staff, and also (where appropriate) for users.

— A plan for creating the guidelines and standards to accompany the method.

— A plan for the phased introduction of the method and tools into new development and maintenance projects.

Figure 5.1 Methods for dealing with resistance to change

Method	*When to use it*	*Advantages*	*Drawbacks*
Education plus communication	When there is either a lack of information or inaccurate information and analysis.	Once persuaded, people will often help with the implementation of the change.	Can be very time-consuming if many people are involved.
Participation plus involvement	When the initiators do not have all the information they need to design the change, and when others have considerable power to resist.	People who participate will be committed to implementing change, and any relevant information they have will be integrated into the change plan.	Can be very time-consuming if participants design an inappropriate change.
Facilitation plus support	When people are resisting because of adjustment problems.	No other approach works as well with adjustment problems.	Can be time-consuming and expensive, and may still fail.
Negotiation plus agreement	When someone or some group will clearly lose out in a change, and where that person or group has considerable power to resist.	Sometimes it is a relatively easy way to avoid major resistance.	Can be too expensive in many cases if it alerts others to negotiate for compliance.
Manipulation plus agreement	When other tactics will not work or are too expensive.	It can be a relatively quick and inexpensive solution to resistance problems.	Can lead to future problems if people feel they are being manipulated.
Explicit and implicit coercion	When speed is essential and the initiators of the change possess considerable power.	It is speedy, and can overcome any kind of resistance.	Can be risky if it leads to people resenting the initiators.

The project manager should agree this plan with system and user management before implementation, and when it is agreed implementation can proceed.

The first component takes the form of a general education programme for development staff and users. At this stage, resistance to the concepts of the new method can be expected. An earlier Foundation Report (No. 25 — System Development Methods) discussed the problems of overcoming this resistance to change and suggested six methods for dealing with it (these methods are summarised in Figure 5.1). An important factor in motivating people to accept the changes required by a new method is the knowledge that their management is fully committed to the method.

The pilot project team members also have an important role to play in overcoming any resistance to the new method, and in acting as 'missionaries' for it. (However, they should bear in mind that, like real missionaries, they may be unfairly attacked, and they should tone down any excessive enthusiasm they may feel for the method.) The pilot team members should be used on as many as possible of the initial projects, acting as the source of knowledge on the use of the method.

Once the method has been implemented, its use must be continually reviewed in order to:

- Improve the guidelines and standards for its use.
- Incorporate any improved technique or tool into the method.
- Ensure that the method continues to meet the needs of the company.

The benefits of using the method should also be monitored. The next chapter describes the benefits that can be expected.

Chapter 6

Monitoring the payback from system development methods

Proprietary development methods are expensive to implement, not just in terms of the cost of purchasing the method and the supporting tools, but in the support costs and the 'lost time' both of users and development staff whilst they are learning how to use a new method. Furthermore, the benefits of using a proprietary method are not all gained immediately and cannot easily be quantified. Some of the benefits arise from the use of techniques or tools, and could possibly be gained without implementing a full method (improved productivity in generating program code, for example). Hence, development managers find it difficult to present a hard financial case for implementing a new proprietary method. Our discussions with organisations who have invested in system development methods suggest that their motivation for so doing was an intuitive belief that methods help to improve the quality, cost, and timeliness of the development process.

One of the prerequisites for generating a good case is that any expected improvements in development performance can be compared with the performance prior to implementing the new method. At the moment, many organisations are not in a position to do this because they do not even have good measures of their existing programming productivity, let alone their performance on other less tangible aspects of system development. Even if there are no immediate plans to install a proprietary method, we believe that management time and attention should be given to measuring current development performance, so that a firm base will exist in the future for selecting methods and justifying the investment in methods.

We were surprised to find that relatively few organisations measure their development performance on a consistent basis, even though several measurement techniques now exist. For example, the Butler Cox Productivity Enhancement Programme (PEP) uses the Productivity Analysis Database System, which relates various measurements to the size of the system being developed. The ten measurements used in PEP are listed in Figure 6.1. Other measurement techniques can also be used, in particular Function Point Analysis, which was described in Foundation Report 47 — The Effective Use of System Building Tools.

Figure 6.1 Development measurements used in the Butler Cox Productivity Enhancement Programme (PEP)

Productivity index (a global measure of efficiency)
Manpower building index (the rate at which additional staff are added to a project)
Elapsed time (planned or actual)
Effort (planned or actual man-months)
Average manpower used on a project
Source statements per man-month
Source statements per month
Total number of errors
Total errors per month
Cost per line of code

Most of the measurements are expressed as a ratio, related to the size of a project.

At the very least, we believe that the following measurements should be made so that targets for improvements can be set when a new method is introduced:

- The rate of achievement of tested function points (or lines of code, for those who prefer to measure development output in this way).
- Error rates during all phases of testing and during initial implementation.
- The time taken to implement a new system.
- The extent to which operational service agreements are not being met because of poor system design or implementation.
- The level of changes and enhancements.
- Maintenance costs.
- Variances between actual expenditure and budgeted expenditure.

Different types of method provide different benefits and thus the payback achieved from their use differs. In general, however, proprietary methods provide two main types of benefit:

— Improved quality of systems.

— Improved control over the development process itself in terms of both cost and time.

One benefit unlikely to be achieved from introducing a method per se is improved productivity. Productivity gains (that is, systems that are developed less expensively and with less effort) typically come from using tools rather than methods.

PAYBACK ACHIEVED DEPENDS ON THE TYPE OF METHOD

The payback achieved from the use of a system development method differs depending on the type of method. All system development methods will improve both the quality of the systems developed with them and the developers' control over cost and time in the development process. However, different types of method lead to different benefits. Figure 6.2 shows the types of payback that can be expected from the different types of method.

For example, methods like Yourdon, which are mainly concerned with the analysis and design phases of development, provide benefits by improving tne quality of systems more than by improving the management control of projects. Alternatively, the major payback from project-management methods like Prism comes from better control of costs and time during the development process.

IMPROVED QUALITY OF SYSTEMS

Many of the organisations participating in our research reported that using a proprietary development method improved the 'quality' of their systems. However, they could not provide quantified evidence to support this view because most of them do not measure system quality, either before or after the introduction of the method.

The definition of quality used by different organisations varies, ranging from finding the minimum of bugs during initial implementation to developing a system that meets the needs of the end users. However, the examples described below show how different organisations have achieved better quality by using proprietary development methods.

STANDARD CHECKLISTS AND DOCUMENTS IMPROVE QUALITY

Many proprietary methods use standard checklists and standard forms to document information about the system being developed. Some methods use a large number of standard forms: one widely used method has 203 different forms that can be used during a system development project. The completed documents form part of the deliverables for each phase of the project. In practice, however, very few commercial organisations use all the checklists and documents provided by a method; instead, they adapt the documentation to suit their specific needs on a project-by-project basis.

Using standard checklists and documents improves the quality of systems because they ensure that development staff and users do not overlook (or

Figure 6.2 Payback from implementing different types of method

	Type of payback		*Type of method*				
					Multiphase		
Payback achieved through	*Improved system quality*	*Improved control of cost and time*	*Project-management*	*Single-phase*	*Analysis/ design*	*System-build*	*Integrated*
Visible plans/progress		✓	▨				▨
Standard checklists, techniques	✓	✓	▨	▨	▨	▨	▨
Graphic techniques	✓				▨		▨
Advanced requirements analysis, etc.	✓				▨		▨
Advanced analysis and design techniques	✓				▨		▨
Advanced building techniques	✓					▨	▨

provide inadequate details about) essential information during the development process, and they ensure that all the necessary information is properly recorded.

For example, by using LSDM a Scottish bank reduced the proportion of development time spent on fixing bugs and on small enhancements from 30 per cent to 5 per cent. The bank believes that this reduction was due to the use of the LSDM checklists and standard documents.

GRAPHICAL TECHNIQUES IMPROVE QUALITY

Graphical documentation techniques, particularly those used during analysis and design, are powerful aids in communication in system development projects. For example, Sodetag-TAI, a French software house specialising in large turnkey software projects, particularly in the areas of metro signalling, load despatch, and message switching, uses SADT for requirements analysis, systems analysis, and specification, and Mach for systems design. The company believes that much of its success is due to its use of these methods. The SADT method is based on top-down graphical analysis where the system is analysed initially at a high level of abstraction, with further levels of detail being added in a logical structured manner.

The use of these methods in Sodetag-TAI is considered so important that two departments specialise in their application. The first department teaches new and existing staff how to use the techniques. The second researches the methods, identifying how to develop and implement them further.

The use of graphical techniques improves the quality of systems by:

- Ensuring that the analysis and design phases are carried out thoroughly and completely, by expanding the top-level general view through successive levels of detail.
- Presenting a large amount of information in a way that is easy to understand (a picture is worth a thousand words), so that it is easier to check the design of the system for completeness and accuracy.
- Providing a basis for automating the analysis, design, and implementation phases.

ADVANCED REQUIREMENTS-ANALYSIS TECHNIQUES IMPROVE QUALITY

Methods based on advanced requirements-analysis techniques, such as prototyping, help to improve the communications between development staff and users and thus improve the quality of systems. Marks and Spencer plc (a major UK retailer) recently commissioned a software house to develop a microcomputer-based system. Marks and Spencer found that using prototyping as a requirements-analysis technique provided the following advantages:

- Users were more involved, and more committed, during the analysis and design phases.
- Using prototyping with an advanced system-building tool (Sourcewriter) allowed the requirements to be developed and analysed by holding discussions with the users, which enabled the users to correct any misunderstandings quickly and to generate new ideas.
- The prototype review, a critical phase in the method used by the software house, allowed all the users of the system to view the prototype system, and to comment on its match with their requirements.

We believe that a 'consensus' environment, where users can freely discuss and agree their needs, is a prerequisite for using prototyping in this way. One of the major factors in Marks and Spencer's successful use of prototyping is the fact that it achieved this consensus.

Other more traditional techniques used in requirements analysis can also help improve the quality of systems. For example, a French car manufacturer found that its use of SDM and Merise helped systems users to formulate their requirements and resulted in fewer rejections of systems at the user-test phase of development.

ADVANCED ANALYSIS/DESIGN AND BUILDING TECHNIQUES IMPROVE QUALITY

Many proprietary methods make it easier to check that applications software is correct because they are based on structured-analysis, design, and programming techniques. The modular construction both of the requirements and of the code enables development staff to check for completeness and consistency more easily. It also allows walkthroughs of the system design and code to be conducted easily. For example, a major multinational oil company found that the use of Information Engineering and a fourth-generation language improved quality by reducing the amount of time spent on maintenance from 70 per cent to less than 50 per cent of the total development effort. The robustness of the systems was also improved.

IMPROVED CONTROL OVER THE DEVELOPMENT PROCESS

Over the years, a continual difficulty with large development projects has been the apparent inability of development staff to measure progress objectively. Users have been told that a system is "90 per cent complete", only to find that as much effort again is required to complete the remaining 10 per cent. This situation is a symptom of the difficulties inherent in managing large development projects. Moreover, the problem is not confined to the systems department; it is also difficult for users to be aware of the development progress that has been made.

However, the progress of a system development project can easily be demonstrated using methods based on project-management techniques. The Caisse National de Crédit Agricole (CNCA) uses Merise for system development. CNCA's experience with this method is typical, with its major benefit being that it allows senior management to track the progress of system development projects easily. The phases of a project are agreed in advance with the users, together with the timescales, objectives, tasks, and deliverables for each phase. Progress can then be compared against these initial plans.

The Department of Health and Social Security (DHSS) in the United Kingdom develops some of the largest systems in the world. Some of the projects are very large, ranging from 500 to 1,000 man-years and costing upwards of $75 million. The DHSS controls these very large projects by using a rigorous system development method (SSADM), which is the standard method recommended for government use in the United Kingdom. The method is supported with automated tools such as Prompt, Diadem, and Maestro. SSADM requires a substantial amount of project documentation to be produced. The deliverables from each phase of development include many different types of standard forms that must be completed by the project team. This very bureaucratic method allows the department to control and measure progress on its projects.

By formalising the development process through the use of methods like Merise and SSADM, the systems development manager acquires, over time, the ability not only to evaluate and demonstrate progress but also to compare like projects. This, in turn, provides the ability to estimate better the time and resources likely to be required to develop a new application system. Indeed, some method suppliers are creating databases of projects that can be used by development staff as an estimating aid.

The use of a proprietary development method also provides another intangible benefit. The very fact that a method, rather than an ad hoc approach, is being used enhances the professional standing of development staff in the eyes of the user community. Many system development managers believe this to be a significant benefit of the use of proprietary development methods.

DO NOT EXPECT PRODUCTIVITY GAINS FROM METHODS ALONE

In our research, suppliers and users agreed that the use of proprietary methods does not, by itself, improve development productivity. Their impact on productivity is achieved in two ways. First, methods based on advanced techniques, such as prototyping, reduce the amount of time required for the analysis phase. Second, the use of methods means that advanced tools, such as system-building tools and IPSEs, can be implemented, and these tools can improve development productivity substantially.

USE PROTOTYPING TO IMPROVE PRODUCTIVITY

Prototyping with advanced system-building tools was discussed in detail in Foundation Report 47 — The Effective Use of System Building Tools. There are three major ways in which prototyping can be used, each providing different productivity benefits:

— Prototyping is used as a requirements-definition technique only, and the prototype forms the basis of the requirements definition. Using prototyping in this way improves productivity by reducing the amount of effort spent in the requirements-definition phase.

— The prototype produced during the requirements-definition phase is used as the basis for the actual implementation of the system, using either a third-generation language or an advanced system-building tool. Productivity is improved by reducing the amount of effort spent in defining the requirements and also by reducing the effort required for the implementation phase (assuming that an advanced system-building tool is used). If a third-generation language is used, the prototype is used merely to define the requirements and is then discarded.

— Prototyping is used as the development method, and the system is developed iteratively, using an advanced system-building tool, with functionality being added as the prototype grows. Using prototyping in this way can improve productivity dramatically because functionally incomplete, but working, prototypes are produced very quickly, and these can be converted rapidly into working systems using the system-building tool. However, few organisations are yet using this radical method of developing systems.

Figure 6.3 Benefits obtained by automating methods with tools

Benefit	Type of tool			
	Project-management	Analyst/designer workbenches	Programming	IPSEs
Improved planning and control	✓			
Improved productivity		✓	✓	✓
Improved interpersonal communication		✓		✓

IMPLEMENT ADVANCED TOOLS TO SUPPORT PROPRIETARY METHODS

Proprietary methods affect development productivity by enabling advanced system-building tools such as IPSEs to be implemented. The benefits of using such tools were discussed earlier in Chapter 3 on pages 14 and 15.

As with methods, the benefits obtained from automating a method with development tools depend on the type of tool. The types of benefits that can be obtained from using different types of development tools are shown in Figure 6.3.

REPORT CONCLUSION

In this report, we have dispelled some misconceptions about the use of proprietary development methods and tools. In particular, we have shown that it is not yet possible to purchase an all-embracing development method that can be used for all types of development project and for all phases of the development process. Furthermore, many tools are designed to support either a particular technique or method. Even the so-called integrated tools do not cover the whole of the development process. Foundation members should review the development processes currently used in their organisation, identifying the activities that cause the greatest difficulties. They should then select the methods that will be required to attack those difficulties.

But methods by themselves will not be enough. The main benefits from using development methods come from improved quality of systems and better control of the management of development projects. Methods by themselves do not improve development productivity, however. Improvements in productivity come from using development tools to automate the activities required by the methods. Indeed, many methods are almost unusable without the appropriate tools. Thus, having selected the methods, an organisation must then choose the development tools that will be used to support the method.

The report has provided advice about how to select the appropriate methods and tools. In practice, however, the benefits actually achieved from using the methods and tools will depend not only on how well they are chosen. They will depend also on how well their introduction is managed. The report has highlighted the need to manage the introduction of a new method as a distinct project and has provided advice about how to do this.

Butler Cox & Partners Limited
Butler Cox House, 12 Bloomsbury Square,
London WC1A 2LL, England
☎ (01) 831 0101, Telex 8813717 BUTCOX G
Fax (01) 831 6250

Benelux
Butler Cox BV
Burg Hogguerstraat 791
1064 EB Amsterdam
☎ (020) 139955, Fax (020) 131157

France
Butler Cox SARL
Tour Akzo, 164 Rue Ambroise Croizat,
93204 St Denis-Cedex 1, France
☎ (161) 48.20.61.64, Fax (161) 48.20.72.58

Germany (FR)
Butler Cox Deutschland Ltd.
Richard-Wagner-Str. 13
8000 München 2
☎ (089) 5 23 40 01, Fax (089) 5 23 35 15

United States of America
Butler Cox Inc.
150 East 58th Street, New York, NY 10155, USA
☎ (212) 486 1760 Fax (212) 319 6368

Australia
Mr J Cooper
Consultants (Computer and Financial) plc Australia
Level 5, 303 Pitt Street, Sydney 2000, Australia
☎ (02) 2870400, Fax (02) 2870450

Italy
SISDO
20123 Milano, Via Caradosso 7, Italy
☎ (02) 498 4651, Telex 350309, Fax (02) 481 8842

The Nordic Region
Statskonsult AB
Stortorget 9, S-21122 Malmo, Sweden
☎ (040) 1030 40, Telex 12754 SINTABS

Spain
Mr Sidney M Perera
Rosalía de Castro, 84-2°D, 20835 Madrid, Spain
☎ (91) 723 0995