

8

EVALUATION AND SELECTION OF SOFTWARE PACKAGES

INTRODUCTION

Computers have already had far-reaching effects in almost every sector of industry. Rapid technological developments in computer hardware has heightened the desire by businesses to purchase mini-and microcomputers. Small computer systems have become more attractive because of the relatively lower cost of acquiring computer hardware. Once a computer is purchased, however, a company must either develop or purchase the required software to process the input data and produce the desired information.

A plethora of software packages are available today. Businesses are tempted to purchase these packages without evaluating the alternative of developing the needed software in-house. To make an informed choice, an evaluation of the costs of the available alternatives is needed, and this requires a well defined plan. One has to take care, since buying a package is often considered a long-term investment, involving a good amount of money. Spending enormous amounts of money on inadequate software packages results in an inadequate or useless computerised system. Issues, concerns and approaches for software package evaluation and selection are discussed here.

TRENDS IN SOFTWARE EVOLUTION

In studying the continuing evolution of computers, it becomes amply clear that while there are dramatic developments in both hardware and software, there is a qualitative difference in these developments. While hardware improvements concentrate on higher speeds at lower costs, with lower energy consumption and in a smaller size, the thrust of software is in making computers easier to use, or user-friendly as manufacturers prefer to refer to this feature. We are also beginning to see more and more reasoning power being built into the software — a trend that is bringing the

terms Artificial Intelligence, Knowledge Based Systems, Expert Systems and Decision Support Systems into the foreground. But the developments in software do not necessarily share the cost-reduction, and quite often imply clear increases in cost.

In making computers more user-friendly and menu-driven rather than command-driven, substantial amount of software needs to be added, and this concept creates an ironical situation where the user finds that while software is becoming simpler and easy to use, it is becoming more expensive. Between the hardware and the computer user, more and better layers of software are being padded with the objective of achieving user-friendliness as well as higher levels of independent reasoning (see Figure 8.1). It is these layers of software which give a purpose as well as ease and functionality to the hardware. These layers are further examined more closely.

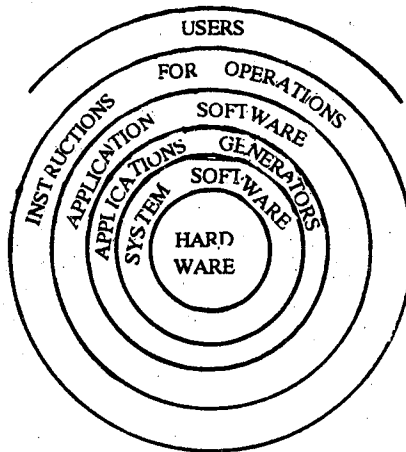


Fig. 8.1 : Layers of Software

1. Systems Software

Systems software is the layer closest to the hardware, and since it is specific to a particular computer model, is supplied by manufacturers along with the computer. In the early days, computer scientists interacted with hardware through a similar software layer, and a high level of skill was necessary to deal with the technicalities of that layer of software. Obviously, a need was felt to build interfaces that would make it easier to use computers. Furthermore, each model of computer required different types of skills and this was becoming an additional hindrance to the widespread use of computers.

2. High Level Languages

High Level Languages emerged as a solution to the problem. Programming languages like COBOL, FORTRAN, etc were developed, with which it was very much simpler to use computers. Further, programmes written for one computer could run on another computer provided an appropriate layer of software was available on both. Over the years, programming languages have gone through significant evolution, and

we are currently using Fourth Generation Languages (4GLs) which are a short step away from the natural languages like English. The second layer of software, then, is that which enables programming languages to operate on the hardware.

3. Application Software

Application software is the set of programmes that are written to get a specific application, e.g. Payroll and Invoicing, to work on a computer. Obviously, these would be different from application to application, and also from organisation to organisation. Thus applications software could be viewed as the third layer. Once an application package has been implemented, what the operating staff (or the end-users, in the case of on-line, multi-terminal systems) would need to know is the sequence of interaction with the system to get the required result. This set of instructions could be viewed as the fourth layer.

THE NEED FOR SOFTWARE

Hardware costs have been tumbling over the years, and this reduction is likely to continue. Software, on the other hand, is becoming more expensive. In the early days of computers, it was common to see computer manufacturers offering software free with the hardware. The situation is changing rapidly and one can expect that a point will come when a buyer will be given the hardware free, if he buys more than a certain value of software. The point of all this is that we need to underscore the opposite trends in costs of hardware and software, and we also need to realise that the real benefits from computerisation comes from the appropriate software, rather than hardware alone.

Software is often the superstructure through which an organisation's goals and objectives may be accomplished. Purchasing software is an important task that deserves proper planning and consideration. It is analogous to the procurement of plant or equipment facilities. Selection of software should be approached with the same care. However, because of the highly abstract nature of software and its function, it can be difficult to follow these steps, particularly when determining costs. Also, writing software is an art still in the embryonic stage. It lacks any generally accepted principles for development and testing, and this increases the difficulty in evaluating the benefits of the package.

Two sources for obtaining software exist. In-house development, particularly of application systems, has been the traditional approach. Now, software packages are emerging as a viable alternative to the in-house development. The perceived costs — both direct and indirect, of in-house development have been the major impetus for the trend in purchasing commercial packages.

SOFTWARE PACKAGES

Various terms exist to describe packages — application packages, application software, standard systems or software packages. Regardless of the terminology used, what is being considered is a package of facilities and services that is designed to

cope with some commercial, statistical or scientific job, for example, stock control, payroll, etc. Technical packages, covering such matters as design and stress calculation, also exist but these are not discussed here.

There are two main categories of software packages. Systems software packages and application software packages. For example the packages Operating Systems (OS), Translators, Simulators and Job Accounting are system software packages while the packages like Payroll, Inventory and PERT/CPM are application packages.

System software packages are used to enhance, extend or drive computer hardware. Although system packages currently represent only 25 percent of the total number of software packages available, they account for approximately 50 percent of the total revenue from software packages. Figure 8.2 illustrates the two main categories of software and their subsets. Utility packages or programmes are a special set of software routines, normally supplied by computer manufacturers, to perform certain standard functions for all users. They fall into two categories, those designed to be used in the operational environment, and those used to assist programmers. Examples of the former are sort programmes, merge programmes and standard printing programmes. In the latter group, examples are store dumps, disk and tape edits and programme diagnostic routines.

Systems Software	Applications Software
- Computer dependent	- Organisation dependent
- Supplied by computer manufacturer	- To be arranged by the buyer
* Operating Systems	* Payroll Systems
* Translators, Compilers	* Financial Accounting Systems
* Utilities	* Accounts Payable Systems
* Data Base Management Systems (DBMS)	* Accounts Receivable Systems
* Job Accounting Systems	* Fixed Asset Accounting
* Telecommunication Systems	* General Ledger Systems
	* Capital Project Accounting
	* Forecasting and Statistical Packages
	* Medical and Health Care Systems
	* Developing and Maintenance Aids for EDP Department
	* Purchasing and Materials Management
	* Inventory Control and Accounting
	* Production Planning and Control

Fig. 8.2 : Categories of software systems

Applications software packages are special-purpose systems designed for business, scientific, medical, academic or other uses. Thus, applications software packages are designed to perform more specific functions, while systems software perform general functions and are typically made available with the purchase of the hardware. Accordingly, we will deal primarily with application software packages:

Typically, a commercial package consists of a computer programme or suite of standardised computer programmes, programme documentation, user's manuals, procedures, instruction booklets, implementation assistance and possibly some formal tuition in the use of the package. The underlying theme of all packages is to provide a more or less readymade system that will cope with individual user's problems.

A software package is a computerised application system, e.g. payroll, accounts payable or bill of materials. Typically, a package is developed by a specialised computer application company for outright sale to organisations needing a specific data processing (EDP) system. One successful software package could be used by several thousand organisations. By purchasing the package, an organisation eliminates the need to design its own EDP system. It is usually safe to assume that a well field-proven package is highly reliable and performs according to the documentation provided by the software vendor. While the decision to buy a software package is analogous to many make-or-buy decisions, buying a software package does not imply elimination of all in-house systems and programming efforts but rather substitution of some of the tasks in the standard systems development and programming life cycle for another set of tasks that, in theory, is easier, less time-consuming and less expensive.

Packages are available from computer manufacturers, specialist software houses, data processing consultants, computer bureaux or ordinary users who are capitalising on hard-won expertise. The concept of packages seems first class in that its spread the development costs over a number of users to become operational that much more quickly, but there are problems. To be usable by a variety of organisations, packages have to be designed for generalised use. They must be able to handle many more situations, types of input and output and processing requirements, than any one user would probably ever need. This tends to make them complicated and difficult to use, at least initially. A more serious problem is the uniqueness of some aspect of the organisation's activities which may not be covered by the package. Nevertheless, if a package can be found that suits some applications it may be an excellent proposition, both financially and operationally.

SOFTWARE PACKAGES AND IN-HOUSE SOFTWARE

Whether a software package or in-house developed software is chosen, the basic goal of the organisation remains the same: to provide a computerised system that meets a definite application requirements. However, these two types of programmes have several distinct differences which are discussed below. Figure 8.3 depicts the differences in acquisition or development of software.

OFF THE SHELF PACKAGES

- * Hardware dependent
- * Proprietary product
- * Inflexible and rigid
- * Least expensive

CUSTOMISATION OF PACKAGES

- * Need experienced manpower
- * Need continual maintenance
- * Some limitations will persist

TAILOR MADE APPLICATION DEVELOPMENT

- * Accepted mode in most countries
- * Complete satisfaction of user requirements possible
- * Takes longer time and effort
- * Most expensive
- * Needs greater user involvement
- * Requires skilled manpower

Fig. 8.3 : Acquisition/Development of Software

Software Package

1. Acquired from an outside vendor, it constitutes a legal contract concerning price and functional capabilities.
2. It is usually installed in multiple organisations, each with the same set of vendor-supplied software.
3. A full set of documentation is available for each management level of the organisation.
4. It is usually field tested prior to public sale.
5. Maintenance is performed by the vendor via a software maintenance contract.
6. It can usually be enhanced over time with DP state-of-the-art improvements.

In-house developed software

1. Design, programming and implementation aspects are developed to suit the needs of one organisation.
2. Maintenance is the responsibility of the organisation's data processing department.
3. Design may or may not consider the latest state-of-the-art software techniques, and is dependent upon the capabilities and expertise of personnel in the organisation.

SOFTWARE DEVELOPMENT COSTS

Cost is the main reason for purchasing a software package rather than developing an in-house system. Developing in-house software system is expensive. Costs include the salaries for programmers, systems analysts and project managers and these costs are increasing faster than the cost of hardware (see Figure 8.4). Developing software is a labour-intensive activity, and it is becoming more difficult to locate and retain experienced, qualified individuals for systems development. In-house software devel-

opment also requires systems teams to use some computer time. Disk space, paper costs and CPU time also add to the development cost. If hardware utilisation is at, or approaching the saturation point, computer time is much more valuable; thus, it becomes much more costly for the systems team to use the hardware.

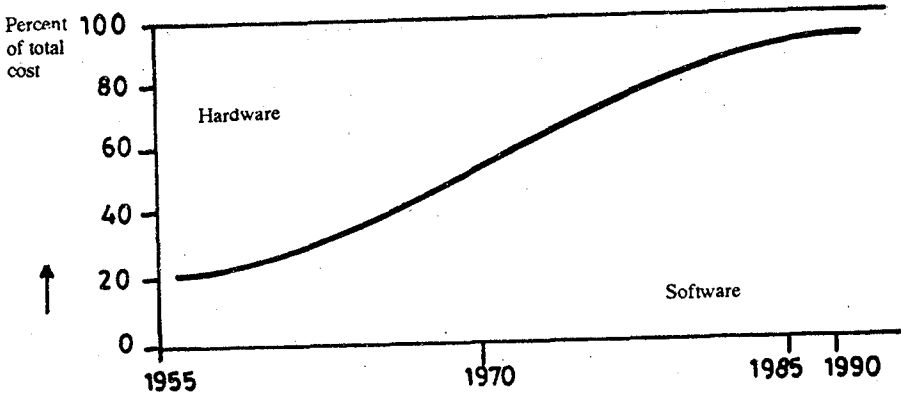


Fig. 8.4 : Trends in Software and Hardware Costs

Studies indicate that (1) software development costs several times the cost of an IC (Integrated Circuit) chip, (2) approximately 70 percent of the total cost of a software product occurs after the product is delivered, and goes into debugging and maintenance, and (3) software maintenance alone takes 50 to 80% of the data processing budget at a typical installation. The increasing importance of software is readily apparent from Figure 8.5. About 80% of the costs was attributed to hardware in the mid 1950s. This ratio is on the verge of reversing by rapidly decreasing hardware costs and by the increasing rate of labour-intensive software costs.

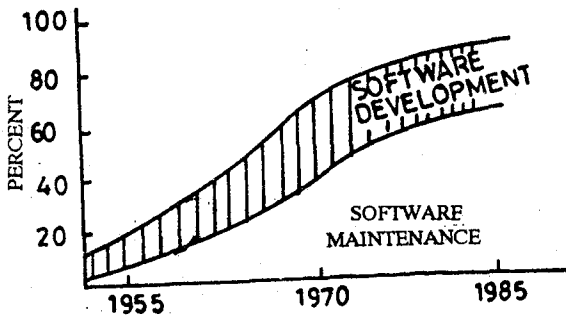


Fig. 8.5 : Hardware Software Cost Trends

Software maintenance plays an important role in computer based systems. Its cost is continually rising and reflects the high labour-intensive effort. The price and performance of hardware have improved by leaps and bounds, but improvement in programmer productivity has been low. This is reflected in the high cost of software development and also gradual increase in the software maintenance cost.

Time can be the other critical cost factor involved in an in-house system. In-house software often is delivered late. If there is a need for virtually immediate delivery, then a software package is more likely to be operative in a short-run time frame.

Currently, there are no precise studies that compare the relative costs of in-house development versus purchasing or leasing a software package. Also, it is important to evaluate the alternatives before making the decision of whether to buy or develop the software.

SOFTWARE PACKAGE COSTS

A software package also involves certain expenditures. Total software costs for package can include the following:

1. Lease or purchase price.
2. Cost of modifying the package to meet internal requirements.
3. Cost of modifying operating procedures.
4. Cost of modifying the operating system so that it is compatible with the package.
5. Cost of modifying or adding computer hardware.
6. Cost of training programmers, systems analysts and other EDP personnel so that the package can be used efficiently.

Many of these costs will be incurred regardless of whether the software is internally developed or purchased (e.g. cost of modifying operating procedures, cost of modifying or adding computer hardware). However, there are many costs unique to purchasing the software package (e.g. the lease or purchase price).

The most frequently encountered cost after the purchase of the package is the cost of modifying a package so that it fulfills the organisation's needs. There usually are unique information and procedural requirements that have to be met. An earlier study on this topic revealed that nearly three-fourths of the application software packages required significant modifications before they were operative. If a software vendor can make the modifications, these costs can be easily identified. If the modifications are to be made internally then determining costs is difficult since part of the cost is an allocation of fixed EDP costs. When modifications are significant, these costs are higher than for internally developed software because the systems team will be less familiar with the software package. This will increase the time, effort and costs for modifying the package.

When packaged software is maintained by in-house programmers, systems analysts and project managers, there also training costs are to be considered. For an in-house system, this training is an integral part of the systems development cycle. When a software package is acquired, time will have to be spent training EDP personnel on the mechanics of the system.

SOFTWARE QUALITY

Before we take up the question of software selection it would perhaps be necessary to know as to what constitutes good quality software package. A good quality software may have a proper mix of the following strongly competing attributes:

- Clear definition of purpose
- Simplicity of use
- Ruggedness
- Early availability
- Reliability
- Extensibility and improvability
- Adaptability and extension to different configurations
- Suitability to configurations of the range
- Brevity
- Efficiency (speed)
- Operating ease for the operator
- Adaptability to wide range of applications
- Coherence and consistency with other programmes
- Minimum cost to develop international standards
- Conformity to national and international standards with respect to character codes, tape formats, languages, etc.
- Clear, accurate and precise user's document

ASSESSING A PACKAGE

To make a realistic appraisal of the suitability of a package is a lengthy process but even so it takes far less time than the investigation, design, programming, testing and implementation of a tailor-made system. The following steps are necessary:

1. Determine the reporting facilities and routine outputs necessary to meet the organisation's requirements.
2. Determine the types and formats of the inputs available within the organisation.
3. Determine all relevant volumes (transactions, reports, outputs, amendments), current and for the future. Step 1 to 3 are normal in any systems investigation and design, whether or not packages are being considered.
4. Determine essential requirements and useful additional facilities.
5. Consider packages available and assess suitability for the application being

considered. The assessment of the package is covered in steps 6-13.

6. Coverage - What is the general coverage of the package? Is it for very narrow area of the organisation's activities or does it encompass many functions?
7. Flexibility - Is the package capable of revision and modification without substantial difficulty? Are a number of exits provided from the standard package for users to create their own additions? Can different reports be generated on demand or reporting requirements be easily inserted into the package?
8. Restrictions - What restrictions on coding, input formats, the organisation's clerical systems, will be caused by the use of the package? What information, if any, will have to be foregone?
9. Assistance - Is full documentation of all aspects of the package provided? What on-site assistance is given during the implementation of the package? Is any formal training given in the use of package?
10. Reliability - What is the source of the package? What do existing users, if any, say about the package itself and the supplier? If the package is not yet developed, what is the record of the package suppliers? Do they have a record of producing high quality, tested material to a definite schedule?
11. Hardware Requirements - What are the memory/peripheral software requirements of the package? Can you use it on your installation?
12. Performance - How well does the package perform? What timings are available? Is a benchmark test possible? What do existing users say about its performance? If package is being developed, does the specification include anticipated performance details?
13. Costs - Will the package be paid for by outright-purchase, lease or by some form of usage charge? How do the costs compare with alternative packages? How do the costs compare with the projected costs of developing a tailor-made system?

WHEN TO SELECT A PACKAGE

There are two predominant reasons for selecting a software package. First, individuals in charge of selecting or using the system are relatively naive with respect to computer programming, computer technology and their specific application. Secondly, the total cost of in-house development (which includes systems analysis, design, coding, testing, implementation and hardware utilisation) exceeds the total cost of leasing or purchasing a software package.

Obviously, in the first instance where the individual or organisation has little experience with programming or computer hardware or even their target applications, they would be well advised to go the software package route since the costs to develop the necessary expertise would be prohibitive. These organisations usually should purchase or lease a complete systems package. So called "turnkey" systems are available

from original equipment manufacturers (OEM). If a user or organisation is in this category, it would be prudent, if not imperative, to hire a consultant to aid in the selection process.

If an organisation is relatively sophisticated with respect to the target application and computer technology, then the comparison of the relative costs of the software packages and in-house development is necessary to get the most from the software package. In doing so, the organisation must make a comparison of the total costs involved in each of the alternatives.

Many of these costs will be comparable regardless of whether the organisation purchases a software package or develops it in-house (e.g., cost of modifying operating procedures). To expedite the analysis, an organisation should focus its analysis on the critical costs.

Software packages are most advantageous when the software is needed soon, the software package can be implemented with only minor modifications and the organisation does not have to maintain the software. Conversely in-house development is preferable when existing packages are not readily compatible with the system, when software packages will require extensive modification by the user organisation and when users are taking total responsibility for software maintenance.

Since time can be critical, the first decision to be made is how soon the software will be needed. If the software will be needed very quickly, or if the hardware is being used at or near capacity for current operations, in-house development will be cost prohibitive. As shown in Figure 8.4 labour costs are increasing rapidly, and to quickly develop the in-house software, an organisation would have to devote a large amount of time and energy to the project, increasing its cost. Also, if the hardware is currently being used at or near capacity, significant indirect costs for in-house software development result because of interruption of operations.

If time is not critical, choosing between purchase and in-house development is not as clear cut. Most software packages require a significant amount of modification before they are fully operational. This is not true of software developed in-house, since it is specifically tailored to specific needs. If the modification must be done by EDP

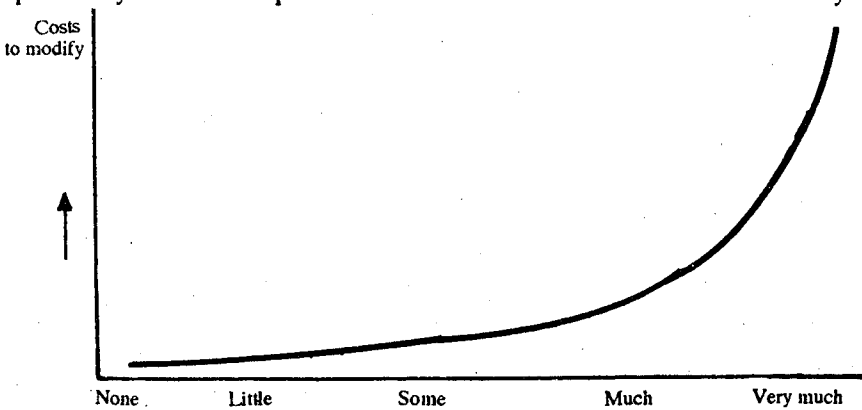


Fig. 8.6 : Cost to Modify Software Packages

department rather than a vendor, the cost of the software package should be increased accordingly. Obviously, as the degree of modification increases, the cost increases and this is not a linear relationship (see Figure 8.6).

The remaining critical factor to consider is the maintenance of the software. If the supplier is to maintain the software, the cost is the agreed upon price which the supplier charges for the service. But, if the organisation plans to maintain the software, there can be hidden costs implicit in that decision. The EDP personnel must be trained so that they can provide efficient maintenance of the system. As a consequence, maintenance is likely to be more costly in terms of the organisation's human resources. However, there is a cost saving that will partially offset the human resources cost. By providing maintenance, the organisation has instantaneous access to maintenance personnel which in the long-run can be quite cost-effective. Figure 8.7 depicts the cost trade-off between acquisition and development of software.

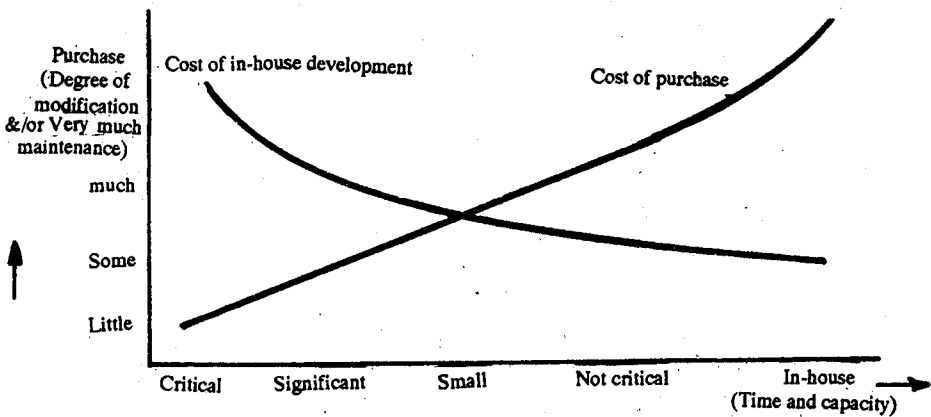


Fig. 8.7 : Relative cost of purchases vs in-house development of software

In general, therefore, packages can be worthwhile in three situations:

1. Where an existing proven package more or less exactly fits the requirements of one of the organisation's applications and is a cost effective proposition.
2. Where an organisation is a new computer user and personnel at all levels are unfamiliar with data processing, then the interim use of packages may be beneficial. Such use will help the organisation gain data processing expertise and refine their requirements' definitions without massive expenditure. Also, the installation is likely to become productive more quickly.
3. The third situation is where an organisation has many of the basic foundation applications operational, using tailor-made systems but wishes to develop the use of more sophisticated techniques. The use of many of these techniques (e.g. linear programming, simulation and PERT) is likely to be only occasional and a package may be the ideal answer, either on the organisation's own computer or at a bureau.

AREAS OF CONCERN FOR MANAGEMENT

The decision to buy/select/make software requires an evaluation process recognising both direct and indirect costs and benefits. Implications relating to in-house development, such as utilisation of data processing staff, work priorities, risk and track record of previous in-house efforts of a similar nature, must be compared to the gamut of factors relating to software package acquisition, e.g. contractual arrangements, vendor selection and goodness-of-fit between user needs and package capabilities. The choice is not always obvious, and further more, it requires a deep consideration of many detailed factors, which are sometimes even harder to determine.

In the make or buy software decision, management is faced with a series of questions that requires understanding of management policies and system implementation goals. These questions provide a focal point for management to decide whether to make or buy the software package. The decision should answer questions such as the following:

1. What are the organisation's objectives and goals pertaining to the specific application in question?
2. Can the organisation develop the application with their current EDP staff?
3. How difficult would it be to recruit qualified software personnel to develop the systems?
4. Are there other uses for additional staff after the new system is implemented?
5. What is the projected cost for in-house development?
6. Are there non-quantifiable costs that should also be considered?
7. How long will it take from the system requirements study to final implementation?
8. Are the new system expectations only for the short range say less than 3 years?
9. Can the organisation's long term needs be identified and included in the new system?
10. What impact will the in-house development have on organisation's resources?
11. What is the anticipated effect of systems maintenance — its costs, duration and resources?
12. Are there software packages that meet the organisation's need?
13. What is the range of cost for acquiring a package that meets identified needs?
14. Do the software packages function on the current hardware?
15. Are there plans to change the hardware configuration? If so, when, and will the package operate on the new configuration?
16. What is the source language of the proposed package?
17. What is the impact of the package on the current hardware?

18. If additional hardware is required, what will it cost and what is the order lead time?
19. What is the performance record of the proposed vendor?
20. What is the performance record of the proposed package?

ADVANTAGES AND DISADVANTAGES OF ACQUIRING A SOFTWARE PACKAGE

In evaluating the alternative method of implementing a software application, the decision-maker should be aware of the advantages and disadvantages of acquiring a software package. Some of these are discussed here.

Advantages

1. Eliminates reinventing the wheel. In many instances, packages provide basic capabilities that are more than sufficient to accommodate an organisation's needs. Often packages have been designed for a specific industry and functional use, i.e. construction industry accounting package or manufacturing bill-of-materials (BOM) software.
2. Often costs less than in-house development. The following table provides a buy versus make matrix for a hypothetical package. The values included in the matrix typify the costs and time required for a package equivalent to a Rs.80,000 software package. As the matrix shows, both investment cost and payback period indicate economic advantages in acquiring a package.

Economic evaluation matrix : buy versus make software.

ESTIMATE	MAKE	BUY
A. Cost of Evaluation Study	Rs.5,000	Rs.15,000
B. Duration of Evaluation Study	1 month	3 months
C. Development Time	18 months	6 months
D. Cost of in-house Development/ Acquisition	Rs.200,000	Rs.40,000
E. Conversion Time	3 months	3 months
F. Conversion Cost	Rs.25,000	Rs.25,000
G. Annual Savings	Rs.100,000	Rs.100,000
H. Projected Payable Time Period	*27 months	10 months

* Sum of (A+D+F)

Rs.8,500 (=Rs.100,000/12 months)

3. Releases software personnel for other projects. Most data processing departments have substantial backlog of user-requested data processing systems. The ratio of personnel and implementation time period is dramatically in favour of the acquisition route. As such, DP personnel become available sooner for other projects. While the ratio in the above mentioned table is 3:1, this ratio can be even higher.
4. Has high reliability and performs according to stated documentation. Packages have been implemented in a large number of installations. Thus, the package is generally field-proven and operates as described in the documentation.
5. Assures that the system is documented. Part of the package acquisition decision involves review of vendor supplied documentation. Unfortunately, when a package is developed in-house, documentation is usually low on the priority ladder. As a result, final documentation is incomplete and too often is not updated during the software maintenance process.
6. Minimises risk usually associated with large-scale system development effort. System development efforts generally have had bad performance records. Management have experience with projects being overdue and over the budget. A package reduces the probability that this situation will occur and places some reasonable constraints on anticipated costs.
7. Provides improved opportunity costs. Since packages are known to be less expensive, the economic advantage offers management the chance to redirect some financial and manpower resources to other projects.

Disadvantages

With all the pluses, software packages can still bring problems which are of following nature:

1. May not adequately meet user's requirements. This is probably the major drawback of acquiring a software package. Too often the organisation's needs are not clearly defined, so the selected software package does not adequately meet processing requirements.
2. Modification of base system results in loss of vendor support. If the selected system requires changes, the purchaser can lose its guarantee by modifying the software code. An alternative is to use the package interface files to develop alternative processing not supplied by the system. These are generally known as Add-ons.
3. Additional hardware may be required. The selected package may involve additional computer resources that would not otherwise be required. Although selection of another package could be an alternative, this choice may not be available.
4. Purchase of packages involves large cash outlay. This substantial investment can be conceived as an added expense since the use of in-house personnel is often

considered a fixed cost. Since many organisations have in-house systems and programming personnel, the problem involves determining how to allocate personnel to the project of highest priority, rather than deciding whether to incur an expense.

5. Implementation responsibility still lies with the organisation and not with the vendor. While software package vendors provide reliable products with some implementation assistance, sole responsibility still lies with the purchaser. Therefore, acquisition of a package does not automatically imply successful installation.

SOFTWARE VENDOR CHECKLIST

Many questions you would ask prospective application software vendors do not differentiate one vendor from another. The similarities between suppliers are not what's important — it is the differences that count. Here are questions you should ask prospective application software vendors before making your selection.

1. Can the application software you are considering be easily delivered in a variety of data processing environments (operating system, teleprocessing monitor and data base) to permit easy migration and allow system software environmental independence?
2. Can the application software be delivered for VSAM and all popular data base management systems, rather than being tied to the application vendor's data base system - one which could be outmoded by new data base products developed by another vendor in this rapidly changing high-technology industry?
3. Is the application software implemented using a "native" or direct approach to the operating system, teleprocessing monitor and data base manager, without the inefficiency associated with "bridges" or the vendor dependencies on a "black box" approach?
4. Is the application product coded in an efficient, widely used and industry-standard language such as COBOL or C rather than a vendor-dependent language unknown to the general data processing community?
5. Can the application product be "tailored" or streamlined to meet the specific functional and operational needs of a company through purchase of a basic package and selected optional features, thereby avoiding delivery of useless code?
6. Does the vendor retain a source version of each customer's uniquely tailored application software for emergency back-up, problem determination and client assistance?
7. Does the vendor have a broad, completely integrated line of application software which can be demonstrated on a single system instead of merely described?
8. Have all of the vendor's products been integrated by design and developed by a single organisation, thus eliminating the need for inefficient interfaces and unknown "black boxes" to tie together unrelated or acquired applications?

9. Does the vendor clearly demonstrate a full commitment to the complex business of application software, rather than offer an incidental addition to its main product line?
10. Has the vendor been in the application software business a minimum of ten years with a successful track record of sustained profits and a strong financial posture?
11. Does the vendor have sufficient capital liquidity to develop enhancements to its product line and to meet unexpected cashflow requirements, and little or no debt to assure its long-term staying power in the event of economic difficulties?
12. Is the vendor's product line not only broad enough to satisfy the product needs of your immediate project, but also broad enough to satisfy the product needs of any subsequent project extensions?
13. Can the vendor provide you with a complete listing and demonstration of integration points between functional systems, as well as plans for specific additional integration points in subsequent releases?
14. Does the vendor routinely provide for a pre-installation planning meeting to prepare the client for the installation steps to be completed?
15. Does the vendor provide educational and training support in the form of public workshops and private, individualised education?
16. Does the vendor provide 24 hours, seven-day-a-week 'hot line' telephone support for service and trouble-shooting?
17. Does the vendor have a formal independent users group which is organised with application interests in view and which meets regularly?
18. Does the vendor enjoy a position of leadership within professional organisations recognised in each application field?
19. Does the vendor provide a comprehensive security system built into the application which dynamically adjusts the system for the user, based upon the capabilities specified in the user's unique security profile?
20. Does the vendor have a maintenance plan available which routinely provides enhancements and corrections on a regular basis?
21. Does the vendor provide source language code for all programmes and routines thereby permitting better user understanding, modification and maintenance?
22. Are all programmes written using top-down, structured techniques with good on-line documentation?

CONCLUSION

To summarise, before your organisation embarks on the development of a tailor-made system, are you sure that no package exists that will cope with the work? When

one considers how many people have independently and expensively solved the same problems in sales accounting, credit control, pay roll and the like, it is enough to make an accountant shudder. When an organisation has a well developed EDP group, the choice between in-house development and purchase of new software is not always clear cut. The advantages of purchasing a software package are that it can be implemented quickly and the initial costs (purchase or lease price) are much lower than the costs to develop the software. However, in-house development provides the advantages of flexibility, compatibility with the existing hardware and lower cost at training personnel. Organisations with a sophisticated EDP department should consider the relative cost of in-house development before looking for a software package to purchase or lease.