



SOFTWARE PACKAGES: STRATEGIC ISSUES

Summary

Fulfilling requirements through using a software package has become increasingly popular. This paper looks at the growth of the software package solution and how it affects business policy, IT policy and business processes. It also considers the advantages and disadvantages of this approach and concludes with the distinction between Request for Proposal (RFP), Request for Information (RFI) and Request for Quotation (RFQ).

Definition

According to ISO/IEC 25051 commercial off-the-shelf (COTS) software products 'are used in an increasingly wide variety of application area and their correct operation is often vital for business, safety or personal applications'. COTS software products are ready-made software packages that are sold off-the-shelf to an 'acquirer who had no influence on its features and other qualities'. ISO/IEC 25051 also tells us that COTS products 'include but are not limited to text processors, spreadsheets, database control software, graphics packages, software for technical, scientific or real-time embedded functions such as real-time operating systems or local area networks for aviation/communication, automated teller machines, money conversion, human resource management software, sales management, and web software such as generators of web sites/pages'.

The ISO definition embraces generalised software (such as spreadsheets and text processors) as well as software for specific application areas (human resource management and sales management). The focus of this paper is on the latter, although elements of the approach can be used for the former.

Historical perspective

There is a strong tradition in computing of bespoke systems development, where in-house analysts and programmers develop systems to meet the specific requirements of an application. In the formative years of information systems development, there was little alternative to this approach because generalised software packages did not exist and the fragmented nature of the hardware market reduced the economic viability of such an approach.

For many years the marketplace was:

- supported by the notion that systems developers had to be part of the company (like production staff, catering staff and cleaners) and located in an identifiable IT department
- dominated by mainframe and minicomputer manufacturers delivering expensive and non-standardised technology. The hardware spend (both for purchase and maintenance) dominated the IT department's budget
- restricted to a limited set of software packages, usually framed to follow legislative requirements (such as integrated accounts) and only available on a restricted number of hardware platforms

- confined to large organisations that usually employed IT staff to produce in-house bespoke computer systems to exactly fulfil the exact needs of users.

However, in the intervening years, a number of trends have affected the marketplace:

- The recognition that IT may be organised as a procurement, rather than development, department. There has been a trend towards outsourcing as companies have downsized to concentrate on their core business. IT is not the only function to be affected in this way. Cleaning and catering, once provided by local employed labour, has often been outsourced by many companies.
- The reduced cost and standardisation of hardware has meant that fewer organisations are tied into a manufacturer's strategy. Software is available across a limited set of dominant platforms and the cost of moving from one platform to another has considerably reduced. Software (or more accurately the people who develop it) is now the most expensive part of systems development. As a result, the cost of software production is closely monitored.
- The passage of time has led to more software products being made available. This has strengthened particular market sectors (for example, there are many integrated accounts packages available in the market) as well as broadening the scope of package solutions. There are packages for golf club administration, patient records, marketing, family tree construction and so forth.
- Time to market has become a more critical part of organisational strategy. Hence supporting information systems are required very quickly and the application software package approach seems to offer the quick implementation required.
- Computer systems are now extensively used in small and medium-sized enterprises (SMEs) that cannot justify the costs of in-house bespoke systems development. Software packages are a much more appropriate way of fulfilling their information systems requirements.

In recent years, the trends has been for software packages to be provided as a service over the Internet. Purchasers do not receive a physical copy of the product, but receive it as a web service.

As a result, many organisations have focussed more on fulfilling their requirements through the purchase of an appropriate software package. The perception is that this is a cheaper, faster and more reliable approach to systems development than developing bespoke solutions. In many circumstances, the task is to find a package that fulfils user requirements or to differentiate between competing packages that all appear to do the job. The advantages of this approach appear so clear that it hardly needs justifying. However, in practice, nothing is so simple.

Relationship of packages to business strategy

Johnson, Scholes and Whittington (2005) have identified business strategy as being about:

- The long term direction of an organisation
- The scope of an organisation's activities
- Gaining advantage over competitors

- Addressing changes in the business environment
- Building on resources and competencies
- The values and expectations of stakeholders.

The ready availability of software package solutions may allow organisations to quickly and easily **extend the scope of an organisation's activities**. For example, standard e-commerce software has allowed companies to quickly move into delivering products and goods over the Internet. In general, software packages have reduced barriers to entry to certain marketplaces. For instance, will-writing software allowed relatively unskilled people to move into an area traditionally dominated by high-cost solicitors.

Software packages may allow companies to **quickly respond to changes in the business environment** and again the growth of e-commerce illustrates this. To remain competitive many companies had to move quickly to offer their products and service over the Internet, to counter moves by their competitors. On a more prosaic level, accounting software packages allow organisations to accurately and quickly comply with legislative changes, such as the way that Value Added Tax is calculated and accounted for.

The efficiency of internal business processes is also of concern to those responsible for business strategy. This efficiency is **significant to stakeholders** and to the **resources employed within the organisation**. Internal analysis of the organisation may identify significant weaknesses and the adoption of a software package may quickly address these weaknesses. For example, problems with budgetary control and reporting may be tackled through the adoption of an appropriate software package. By using the competencies of others, problems within the organisation can be tackled. Control of cost and cost reduction may also be an important element of the agreed strategic direction and packages may again have a role to play in this. For instance, a failure to automate very simple, routine processes may lead to higher costs and error rates than appropriately automated competitors.

However, gaining **advantage over competitors** may be more difficult if software packages are used, because their functions and facilities are available to the organisation's competitors. Any competitive edge will have to be developed through combining the packages in a unique way. It may be that competitive advantage has to be sought elsewhere, for example; in after-sales service, quality or in the competencies of the employees of the organisation. This issue is considered again when the advantages and disadvantages of the software package approach are considered.

Relationship of packages to IT strategy

As Paul Harmon has pointed out, in the 1990s many companies installed off-the-shelf application packages from a number of vendors such as SAP, Oracle and PeopleSoft. Such vendors initially stressed that they sold application software that performed certain common tasks that companies required, like those in accounting, inventory and human-resources. Later, in response to the widespread interest in business process improvement, these companies began to reposition themselves. They developed templates or blueprints that showed how groups of the modules could be linked together to create business processes. People began to refer to these groups of applications as Enterprise Resource Planning (ERP) applications and recently some have added Customer Relationship Management (CRM) applications to the portfolio. In essence, the vendors have introduced workflow-like applications that allow companies to specify or modify the flow of control from one module to another (Harmon, 2003).

In sales literature, vendors often refer to their applications as ‘best processes’. They argue that they have developed their modules from discovering what worked best in several companies. Hence they represent very efficient ways of handling the processes and activities they support. However, in contrast, Paul Harmon argues that the modules represent ‘average processes’ and that, once a company decides to use such modules, then their process is essentially the same as competitors who are using the same modules. No competitive edge can be gained from such applications.

Harmon believes that ERP applications represent a reasonable approach to improving a wide variety of business processes. If the processes are easy to automate and add little value to the overall business then there is no reason why the organisation should not rely on efficient average solutions and focus its energies instead on core processes that do add significant value. After all, managing the payroll deductions or an office inventory database are enabling processes that need to be done but do not add anything to the bottom line. One leading advocate of this approach is Thomas Davenport, one of the consultants who kicked off the business process re-engineering movement in the early 1990s. He argues that a packaged application approach allows companies to rapidly integrate and improve their software systems and to integrate diverse processes. Hence, their processes can be rapidly brought up to speed.

The problem comes when companies try to use such applications for tasks that are not routine and decide to tailor the software to make it fit with the way that the company does business. If a company acquires an off-the-shelf package and then decides to tailor it, the value of buying an off-the-shelf application diminishes rapidly. The usual benefits of reduced cost, speed to market and high quality are rapidly undermined by the cost of tailoring and maintaining the tailored solution. Harmon suggests that ‘if you find yourself considering ERP applications and simultaneously planning to make lots of modifications you’re probably making a mistake. If the process is really a routine process and adds little value it is probably better to change your organisation and use the application in its standard version. If you really can’t live with the vanilla version of the application, then you have to ask yourself if you really want to buy an ERP application in the first place’.

In effect, ERP packages provide the business process solution. Existing processes need to be modified to accommodate the solution. The business analyst needs to understand the ‘gap’ between the current business processes and the processes supported by the ERP solution. This gap may be addressed in two ways:

- 1) Changing the current business process to those supported by the ERP solution. This requires the business analyst to be a significant change agent, producing detailed processes, training plans and convincing employees of the benefits of the new approach. Employees responsible for managing and carrying out the current business process have to buy-in to its proposed successor.
- 2) Changing the packaged software to fit the business process or commissioning bespoke software that bridges the gap between the current process and the vanilla version of the software package. The problems of tailoring the package have already been touched upon. Commissioning software, often developed in-house, to bridge between the package and the process might be a reasonable compromise but still raises cost, time, quality and maintenance issues.

A good example of a company that used such packages to reorganise their business processes is the US subsidiary of Nestle (quoted in Harmon). It decided to standardize all of the major software systems in all of its divisions. A key stakeholder team was setup to manage the whole process. It decided to standardize on five SAP modules - purchasing, financials, sales and distribution, accounts payable and accounts receivable. They also decided to implement a SAP compatible supply chain module.

As the various applications began to roll out to the divisions, the stakeholder team managing the entire effort began to get lots of unpleasant feedback. In hindsight, they recognised that they had completely under-estimated the problems involved in changing the division cultures or modifying established business processes. By 1999 the rollout was in serious trouble. The workers did not understand the new modules and they did not understand how the outputs they were getting from them would help them do their jobs or manage the processes they were responsible for.

The team agreed that it was relatively easy to install modules but that it was **‘very hard to win the acceptance of the people responsible for assuring those processes operated correctly’**. The team began to focus on modelling processes and defining process requirements before creating a plan to install the modules. Several installations were delayed for months or years to accommodate groups that were not prepared for the process changes required.

The person responsible for much of the project claimed that if she had to do it over again she would **‘focus first on changing business processes and achieving universal buy in and then and only then on installing the software’**. Most large organisations have found that installing ERP software takes much longer and is more painful than they had hoped. The problem lies in the fact that the applications are not solutions, they are a tool to use in changing the business processes. The transition must be conceptualized as a business process transition and guided by business managers. The ERP application must be installed as part of the overall business process redesign effort, not as an independent activity. Used in such an appropriate manner, ERP applications offer a powerful tool to aid business. Used inappropriately, ERP applications can cripple both the processes and financial stability of an organisation. The Nestle example is discussed in full in Harmon (2003).

Relationship of packages to business processes

Figure 1 (taken from Paul Harmon, 2003) identifies strategies for business process improvement based upon examining the complexity and strategic importance of individual processes.

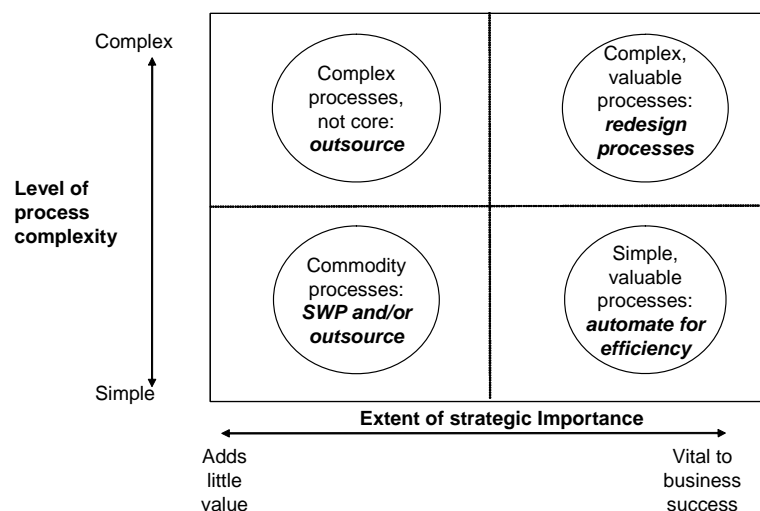


Figure 1: Strategies for business process change

The matrix in figure 1 is very useful when deciding upon the most appropriate approach to redesigning processes. For example, where a process such as payroll is concerned very few organisations would undertake an expensive process re-design project. In this case, the most typical approach is to purchase an off-the-shelf software package or even outsource the entire payroll process to a specialist organisation. As identified clearly in the matrix, organisations need to focus their processes improvement efforts on the processes that deliver competitive advantage because they differentiate the organisation from their competitors.

Processes in the lower left quadrant are largely simple processes that must be done but add very little to products and services offered by the organisation. Payroll (already mentioned above) is a typical example. Processes in this bottom right hand quadrant are straightforward, relatively static commodity processes that are ideal for off-the-shelf solutions. Processes that fall in the lower-right hand quadrant are again relatively straightforward and static but they add value to the products and services offered by the organisation. Again, these are candidates for off-the-shelf package solutions. The processes in the upper-left hand quadrant are complex processes that have to be done but add very little to the value of the organisation's goods and services. These complex services (such as legal advice perhaps, or funds investment) are best outsourced to specialists who have expertise in this area. Finally, the top right hand quadrant for complex processes that add value to the company's good and services. These are often hard to automate completely and it is where the company should focus its business improvements.

Legal issues and legal requirements

The organisation may have internal rules about how procurement should be undertaken. These rules are usually designed to ensure that the organisation receives best value for money spent. Larger organisations will also have rules that should prevent conflicts of interest between managers of the organisation, procurement and suppliers. These rules will define a fair tendering process.

In public sector organisations, tendering rules are usually defined by government or supra-government bodies. In the European Community, there is legislation requiring all public-funded tenders above certain thresholds to be advertised in the Official Journal of the European Union (www.ojec.com) and there is also a standardised tendering process.

Here is a statement from the University of Edinburgh website which is typical of public funded bodies:

As a publicly funded body, the University must abide by EC Procurement Legislation. This was set up in order to ensure open and fair competition amongst the member states of the European Union. The basic principle is that all suppliers must be treated equally and fairly no matter which country they are based in. The University is also publicly accountable to the HM Treasury to show that we are ensuring that best value for money is obtained from public funds.

On the East Sussex County Council website there is specific information about tendering and thresholds (www.eastsussex.gov.uk):

Companies applying for Council business must follow the appropriate tendering procedure. The higher the estimated value of the goods, services or works, the more rigorous the tendering procedure.

If the business opportunity is very straightforward and the value is below the Council's threshold for inviting tenders, then a quotation will be requested as described below.

If the business opportunity is valued above the Council's tender threshold, then the Council will normally ask suppliers to complete a pre-qualification stage. The pre-qualification stage helps the Council check that companies are legally, financially and technically sound. Only those who pass the pre-qualification stage will be invited to submit a tender.

Thresholds for goods and services

Goods or services under £1,000

The Council will normally obtain a quotation from a single supplier.

Goods and services between £1,000 and £5,000

The Council will normally obtain a written quotation from at least one supplier.

Goods and services between £5,000 and £50,000

Interested suppliers will be asked to complete a quotation form. The Council requires written quotations from at least 3 suppliers.

Goods and services between £50,000 (tender threshold) and £139,893

These contracts will be formally tendered and advertised on the South East Business Portal and in the local, national and trade press if appropriate.

Interested suppliers will be asked to complete a below EU level pre-qualification questionnaire.

Suppliers who pass the pre-qualification stage will be invited to tender or negotiate under Council procedures.

Goods and services above £139,893 (EU tender threshold)

These contracts will be formally tendered and advertised on the South East Business Portal, in the Official Journal of the European Union and in the local, national and trade press if appropriate.

At the European tender threshold, interested suppliers will be required to complete an EU level pre-qualification questionnaire. Suppliers who pass this pre-qualification stage will be invited to tender or negotiate under European procedures

Standards

IEEE 1062-1998 is the Recommended Practice for Software Acquisition. It describes a set of quality practices that can be applied during the software acquisition process. It offers a nine step process in software acquisition

1. Planning organisational strategy
2. Implementing organisation's process
3. Defining the software requirements
4. Identifying potential suppliers

5. Preparing contract requirements
6. Evaluating proposals and selecting the supplier
7. Managing for supplier performance
8. Accepting the software
9. Using the software

It is supported by a number of checklists and appendices.

ISO/IEC 25051 describes an international standard for *Software Engineering – software produce requirements and evaluation (SQuaRE) – requirements for quality of commercial off-the-shelf (COTS) software product and instructions for testing*. This international standard is concerned with providing the user community with confidence that the COTS software product will perform as offered and delivered. It does not deal with how the software is produced because the quality system of the supplier is outside the scope of the international standard. It provides third-party certification that provides customers and potential customers with confidence in the software. It provides a mechanism for suppliers to provide customers with confidence about their product. The standard gives detailed instructions to suppliers about how they can ensure that their product conforms to ISO/IEC 25051 so that they can advertise that it does so.

Configuration and customisation

Tailoring and amending the software package is often suggested when the package fails to completely meet all the user's requirements. There are a number of approaches to this. The first option is to ask the software house to make the changes and to pay accordingly. However, this begins to reduce the advantages of the package. It incurs cost and delay and, because it involves program specification, design and testing, it can also reduce the quality of the product. It also removes the "try before you buy" element from part of the product and it is unlikely that bespoke documentation and training will be produced.

This approach also assumes that the software house is willing to tailor the product to meet an organisation's specific requirements. In many instances the software house is not prepared to make such amendments. The reason for this is that the company wishes to offer a standard product. Tailoring a particular implementation creates specification, programming and testing problems and costs and it is unlikely to be very profitable. However, their reluctance is primarily due to the problems of controlling releases and ensuring that future upgrades (containing new functionality and fault fixes) work properly with both standard and tailored releases. This latter issue is a major problem because most software houses want to reduce the number of versions they support, not increase them. Changes are controlled more easily if they can be rolled into the standard version of the software.

An alternative to software house amendment is the purchasing and amendment of source code by the Customer Company. In this approach, the customer buys the source code of the product at a certain instance in time and then undertakes all subsequent amendments and upgrades himself. This removes most of the advantages of the software package approach. The product now becomes a bespoke solution – but a solution where internal developers are (at the outset) unfamiliar with the construction and operation of the programs and data structures. Consequently, there is a large learning curve in the early stages of source code acquisition. Two other problems must be recognised and they both concern support:

Errors due to coding are no longer fixed under a support contract, even if those errors occur in code unaffected by bespoke changes.

Future upgrades of the native software package cannot be reliably integrated into the bespoke solution. The specification of the changes may be used in producing bespoke amendments.

System integration issues

The need to integrate software package solutions with other software and other systems is increasingly important. The ability for the software to pass and receive data and messages from other parts of the corporate solution has to be recognised in the evaluation of alternative packages. The architectural foundation of software packages is becoming increasingly significant in product selection. Organisations do not wish to be tied into proprietary solutions, particularly with the increasing need to link systems to suppliers and customers (often via the Internet). System integration will also form an important part of the testing and implementation of the selected package solution.

Advantages and disadvantages of the software package approach

This section summarises the advantages and disadvantages of the software package approach to information systems development.

Advantages

Cost

The most quoted advantage of the software package approach is reduced cost. The purchase of a software package is perceived as significantly cheaper than developing a bespoke alternative. In a bespoke system the cost of systems development is borne completely by the organisation commissioning the system. In a software package solution, the cost of the systems development is spread across all the potential purchases of the system. For example, a building company estimating, accounting and job control system that currently costs £2,000 to purchase, actually cost £600,000 to develop. This cheapness is usually an important factor in deciding to pursue the software package approach.

Time

Bespoke systems development needs to be tightly specified, designed, programmed and tested. These parts of the development lifecycle are very time-consuming and during this period requirements may change, so complicating the process even further. The software package is a product that already exists. It can be purchased and implemented almost immediately. There is no requirement for design, programming, unit and system testing.

Quality

A further perceived advantage of the software package solution is hinted at in the previous section - the absence of unit and system testing. The software package is a proven product that has undergone systems testing (in development) and user acceptance testing (by the users who have already bought and used the package). Hence the product should be relatively bug-free, as well as fulfilling most of the functional requirements of the application. The implementation should not be affected by the programming errors and misconceptions that can bedevil bespoke systems development.

Documentation and training

In the software package approach the documentation can be inspected and evaluated before purchasing the product. The documents (such as user manuals and HELP systems) are usually of high quality because they represent an important part of the selling process. In contrast, the documentation supporting a bespoke systems development is not available until very late in the lifecycle and is often sub-contracted to users who do not have the time to do the job properly.

A similar principle applies to training. Prospective purchasers can attend a course prior to buying the product and so further evaluate the suitability of the package. Similarly, economies of scale allow the software vendors to produce and provide high quality training courses, supported by professional trainers.

Maintenance and enhancement

Software products are usually supported by a formal maintenance agreement. Although this agreement costs money, it usually provides:

- unlimited access to a Help desk, where experts can sort out user problems
- upgrades to the software that correct known faults and also include new functionality defined and agreed with the user community.

The cost of this support and enhancement is again spread across a number of users and so can be offered relatively cheaply to each individual customer. The cost of providing such services would be extremely expensive if they were borne completely by an organisation commissioning a bespoke development. The upgrade issue is particularly significant to organisations purchasing accounts and payroll packages. The functionality of such systems is affected by legislative changes made by government. These changes are frequent and unpredictable. It is comforting for the customer to know that all amendments are covered by the agreed software contract.

Try before you buy

This is perhaps the most significant advantage of the software approach – the ability to examine the product in detail before purchasing it. This is clearly not possible in the bespoke approach to systems development where the product is not ready until the end of the project. The evaluation of the package can be assisted if it can be borrowed (or rented) for a trial period and used in the target hardware and software environment. This can be supplemented by visits to actual users (reference sites) where the operation of the package can be observed and user comments and experiences documented.

However, it must be recognised that the ability to "try before you buy" places the initiative with the user - not the supplier. Hence the fit to user requirements must be ascertained and confirmed by the customer - not the supplier.

Disadvantages

Ownership

In the bespoke systems development approach, the ownership of the software usually resides with the purchaser - the customer, not the supplier. This is particularly clear if the development is undertaken "in-house", because the ownership of the code clearly resides with the organisation, not the IT department or individual programmers. Even if an external software house produces the code, the contract usually specifies that the source code belongs to the commissioning agent (the customer) and not the supplier.

In the software package approach, the ownership of the software usually remains with the supplier. Customers are licensed to use the product, but they never own it. This ownership issue has a number of implications:

1. The supplier decides the future development of the package. Hence, future functionality is not in the control of the customer - although of course they can lobby for certain features to be included in future releases.

2. The software supplier can make decisions about the ownership and support of the product. These decisions may have far reaching effects on current customers. For example:
 - The software supplier may decide to withdraw support from earlier versions of the package. Hence customers may be forced into unnecessary (and potentially expensive) upgrades. This may involve hardware upgrades.
 - The supplier may completely change the basis of the licence, leading to significant unforeseen cost increases for customers.
 - The software supplier may decide to sell their product to a third party. Individual customers may be unnerved or inconvenienced by such a move. For example, a few years ago a popular DBMS (database management system) was sold to a large industry player with a reputation for aggressive customer relations. Many customers of the DBMS were worried by this move (despite assurances from the new owner) because they were more comfortable with the "laid-back" approach of the former owner. Some began to make plans to move their systems to a rival product.

The key issue here is that the software purchaser has little control over the future direction and ownership of the product they are buying. This is not the case with a bespoke development.

Financial stability of the supplier

Internal Information Systems (IS) departments do not go out of business. However, external software suppliers are subject to the vagaries of management and the markets. There is a risk that they may go out of business, or experience financial problems that affect the quality of their support and development services. It is possible to reduce these risks (through escrow agreements) but the disruption likely to accompany the enactment of such an agreement should not be underestimated.

This problem is accentuated when the software solution is provided as a web service. The loss of service is instant. If the customer has a physical copy of the software he is at least able to continue using it until some alternative can be arranged.

No competitive edge

Many organisations claim that they use (or wish to use) IT and IS as a competitive edge in the market place. They develop bespoke systems to give them that edge. In the software package approach, the software solution (or product) is open to all competitors and potential competitors. It is difficult to see how such a solution can provide a competitive edge, as all potential competitors have access to that solution.

Failure to fit requirements

One of the most commonly claimed disadvantages of the software package approach is the inability of the product to fit all (100%) of users' requirements. This means that either users have to make compromises and accept that they will not get all the functionality they require, or tailored amendments will have to be made to the software product to deliver the required functionality.

Whichever way is chosen, it is clear that most software packages do not fulfil all the user requirements defined for a particular application. Furthermore, they often include facilities and functions not required by a particular user, which only serves to confuse when the product is implemented into the organisation. In contrast, the bespoke solution should completely fulfil all the user's requirements and, if it doesn't, will be amended until it does.

Lack of legal redress

In a bespoke development, the ultimate failure of the system to fulfil the user's functional requirements can be resolved (usually in the favour of the customer) by law. Clearly this last resort is inappropriate if the system has been developed by an internal IS department, but it is an option if the system has been developed by an external software house. Legal redress is well documented in a number of high profile cases.

However, the legal responsibilities of a software package provider are more complex. The licensing agreement is defined in favour of the supplier. In that agreement there is usually a clause that states that the package may not support the functional requirements of the customer – and it is the customer's responsibility to ensure that it does. Unfortunately, many customers do not take that responsibility seriously and are unable to properly assess whether a particular package supports their needs – or 80% of their functional requirements – whatever 80% means!

The changing nature of requirements

There is plenty of evidence to show that requirements change during the lifetime of a system. These changes are due to a number of factors:

- Users change. New managers arrive who have a different perception of requirements and the business process. They demand new output reports to support their particular management information requirements.
- The business changes. The business may decide that it wants to operate in a different way. This may be led by new product and marketing initiatives or may simply be a reaction to changes in the business environment that force the company to re-think its organisation, products and services.
- Changes due to:
 - Actual experience - using the software may lead to a realisation that functional requirements were not quite correct.
 - Emergent hardware and software technologies - bespoke systems can usually be amended to reflect these changes. However, software packages may or may not. The problem is still one of control. It is not possible for the software supplier to predict or to cope with changes in a particular business implementation. Some of the changes may be incorporated into a future release but, at the outset, there is no guarantee of this.

The main point is that users normally evaluate potential packages against the current requirements of current users and this may lead to products quickly becoming inappropriate in dynamic organisations and market places.

The purchasing of source code may reduce programming effort, but the time taken to confidently understand the operation of the software often means that such time (and cost) savings are marginal. There is also no guarantee that the underlying design is flexible enough to support the changes that are likely to be required in the future.

In general, tailoring (changing the source code) the software package is not good practice and the alternative (changing the business practice to reflect the package) is cheaper and more reliable.

Risks of the software package approach and customisation

All enterprise and projects incurs risk. Risk management is concerned with identifying potential problems and eliminating or reducing the potential damage, where possible, or monitoring effects to minimise disruption. Failure to adequately manage risks will threaten the success of the project

Risk management is the responsibility of the Project Manager because she/he is responsible for the success of the project. A well-planned approach to risk control allows the Project Manager to concentrate resources in those areas where risk is high and reduce risks to acceptable limits.

Risk assessment and management must be conducted throughout the project life cycle. It is usually impossible to eliminate all risk but it is possible to manage projects in a way that recognises the existence of risks and prepares in advance, methods of dealing with them if they occur.

The risk management process requires that each risk is assessed and measures formulated to avoid it occurring (avoidance actions) or minimise its effect should it occur (mitigation actions). Both need to be considered because avoidance measures may fail.

For example:

Risk	Avoidance actions	Mitigation actions
Are there multiple suppliers of hardware?	Choose one supplier	Choose a prime contractor and give them responsibility for sub-contracting
Excessive use of inexperienced staff?	Only use experienced staff.	Impose short personal deliverables

As a project proceeds, the nature of risk changes. Old risks disappear and new ones appear. Consequently, risk management is a continuous process and so there needs to be a procedure to regularly review and reassess risks. Furthermore, it is essential that all risks are owned by someone who has sufficient authority and resources to do something about the risk. In some instances risk are assigned at too low a level (the owner understands the risk but cannot do anything about it) or too high a level. In this case, the owner has the resources and authority to do something about the risk but does not understand the risk or give its solution sufficient priority.

The disadvantages of the software package approach are potential sources of risk to the project and the organisation. Suggested avoidance and amelioration actions for four major areas of risk are shown in Table 1 overleaf.

Risk	Avoidance	Mitigation
Ownership	<ul style="list-style-type: none"> • Ensure that the software package has an active independent user group prepared to lobby against draconian changes • Ensure that user group issues are heavily weighted in the appropriate evaluation matrix • Place or amend clauses in the contract about the support of previous versions of the software 	<ul style="list-style-type: none"> • Set funds aside to move application to a competing package or an in-house solution • Play an active role in the user group, or establish one if it does not exist • Take legal advice about any detrimental plans of the company in the context of the terms of the licence agreement
Financial stability of supplier	<ul style="list-style-type: none"> • Ensure that financial issues are heavily weighted in the high level evaluation matrix • Suggest <i>Imperatives</i> concerning years established and/or financial profitability • Purchase from global companies as in the old adage “nobody got sacked for buying from IBM” 	<ul style="list-style-type: none"> • Enact escrow agreement and set funds aside to implement such an agreement if the software house fails • Maintain in-house teams competent in the language the software is written in • Set funds aside to purchase the software house should it get into financial trouble
Failure to fit requirements	<ul style="list-style-type: none"> • Ensure that requirements are properly identified before the ITT is issued • Ensure that all stakeholders take place in the functionality compromise (the defining of the 80% fit!) • Ensure that all stakeholders confirm the functional fit of the package prior to the order being made 	<ul style="list-style-type: none"> • Change the business processes to fit those of the package • Produce a “bolt-on” bespoke application to fulfil the requirements that are not being met • Lobby for the requirements to be included in a future release of the software • Tailor the package to fit the requirements
Changing nature of requirements	<ul style="list-style-type: none"> • Ensure that the software has large elements of configuration and weight this heavily in the appropriate matrix • Ensure that product is developed in a flexible way using software that permits easy changes and expansion • Ensure that new users “buy” in to the current business process and do not request change for change sake 	<ul style="list-style-type: none"> • Produce a “bolt-on” bespoke application to fulfil the requirements that are not being met • Lobby for the requirements to be included in a future release of the software • Tailor the package to fit the requirements • Set funds aside to purchase the source code of the package and for future bespoke development of the system

Table 1: Software Package approach – risk avoidance and amelioration

Distinguishing between RFP, RFI and RFQ

A request for information (RFI) is a way for potential buyers to determine the feasibility of their planned initiative. Although the RFI includes requirements, suppliers are requested to respond to the reasonableness of these requirements and to highlight problems and to suggest other features (not identified by the buyer, but available in the supplier’s solution) that might be of interest to the buyer. The RFI is a way of establishing information that will eventually find its way into the RFP (request for proposals) or invitation to tender (ITT).

Organisations tend to use RFIs when they are still unclear about requirements, scope or the technical or financial feasibility of the whole proposal. The RFP may eventually be developed from the RFI. RFIs allow suppliers to proactively engage with the potential customer and to shape the eventual RFP. It is also common for buyers to only send RFPs to organisations that have responded to the RFI. Consequently, response to the RFI helps determine which suppliers will be on the circulation list for the RFP.

RFQs (request for quotation) are usually sent out when the nature of the requirements is very clear and the supplier can have very little element of doubt about the buyer's needs. For example; the supply of desks, stationery, components, desktop computers etc, can all be dealt with by a RFQ. The goods offered by the competing suppliers cannot easily or significantly be differentiated and so there is no need to hold demonstrations. Many RFQs are issued over the Internet and often price determines the winner. Unfortunately some organisations have used RFQs when the RFP process would have been more satisfactory. Consequently, they have bought goods or services on price, not realising that they were not comparing like with like.

Request for Proposals (or Invitations to Tender) form the basis of the package selection process. Other papers describe a process that should be followed to confidently select a solution from competing software solutions that offer different functionality, from a range of suppliers using a variety of technical design solutions and architecture. RFPs are used where we wish to seek 'best fit' between our requirements and the delivered solution.

Summary

This paper has defined what is meant by a software package and has considered the relationship of these packages to business strategy, IT strategy and business process redesign. It has also identified some perceived advantages and disadvantages of the approach and introduced elements of project management which are appropriate to package evaluation and selection. Finally, it has distinguished between RFP, RFQ and RFI. The contents of the RFP and RFI are considered in detail in the companion paper *Requests for Proposals*.

References and further reading

- *Exploring Corporate Strategy (8th Edition)*, Gerry Johnson, Kevan Scholes and Richard Whittington, FT Prentice Hall, ISBN 978-1405887328
- *Business Process Change*, Paul Harmon, Morgan Kaufman, ISBN
- *The Coming Commoditisation of Processes*, Thomas Davenport, in *Harvard Business Review*, June 2005
- ISO/IEC 25051, British Standards Institution
- IEEE 1062-1998, Institute of Electrical and Electronic Engineers