



FLETCHER IT SOLUTIONS

Essential Guide Series

Successful Custom Software Implementation



Microsoft
CERTIFIED

Partner

Contents

	Page
1. Introduction	2
2. Purchase options	3
2.1 A little history	3
2.2 Packaged software	3
2.3 Custom software	5
2.3.1 Tailored software	5
2.3.2 Bespoke software	5
3. Guide to developing successful custom applications	8
3.1 Business process analysis	8
3.2 Functional and physical requirements	8
3.3 Development	9
3.3.1 Phased development	9
3.3.2 Iterative development	9
3.4 Implementation and deployment	10
3.5 Training	10
3.6 Support and maintenance	11
4. Conclusion	12
5. References	13

1. Introduction

The availability of low-priced application software has spread across all industry sectors, and now covers nearly all business processes. This has led many organisations to believe that they can solve all their problems by using only packaged software.

While there are many areas in a business where packaged software is the right approach, it is not the universal panacea that software manufacturers would have you believe.

Where you have business processes that are particular to your company and provide you with a competitive advantage, it is highly unlikely that any packaged software can provide all the benefits you are hoping for. In this case custom software is the right and only choice.

Many organisations are put off custom development because they believe it is expensive and the end results are not always guaranteed. With this essential guide, we aim to provide you with the facts, so that you can ascertain when packaged software will provide the best solution, and when it is best to pay for custom development.

To try and demystify the process of custom software development, we also describe the steps you should expect your developer to follow in producing a solution that meets your needs and helps your business to grow.

Comparison of packaged and custom software

	Packaged	Custom
Costs	Initially lower	Initially higher, but can work out more cost-effective in the long run
Time	Savings on implementation	Savings on initial training and day-to-day use
Functionality	Feature rich and over complex Stand-alone	Exact fit with requirements Integrates with other software
Maintenance	No control over future	Enhanced support
Enhancements	No input	Total control
Business processes	Compromise – will not fit unique processes	Exact fit to existing processes Easier to use
Ownership	Only licence to use	Code ownership (Escrow)
Competitive edge	Does not enhance	Can significantly enhance by automating processes that make you unique

2. Purchase options

2.1 A little history

In the early days of IT the only realistic approach for many organisations was to develop a software application from scratch. This was because the lack of standardisation in both hardware and software platforms made it uneconomical for commercial application developers to offer many off-the-shelf packages.

With the advent of the PC in the early 80's the pendulum swung firmly in the other direction, with many packaged applications available on the standardised hardware and software platforms that have developed over the last 20 years.

Although these packaged applications have provided productivity gains by enabling organisations to reduce costs and improve efficiency, many organisations are now realising the limitations of such an approach. This is because off-the-shelf packages are designed for the most common business processes, and consequently will rarely fit the unique and precise needs of a particular organisation.

Coupled with this has been the reduced cost of development of customised applications. In the past application development was limited to large companies, but now they are within the price range of most organisations.

Business software applications can now be acquired in one of 3 ways:

1. Bought off-the-shelf in a package and used as is
2. Custom software:
 - As above but tailored to meet specific requirements
 - Developed from scratch (i.e. bespoke)

We are now going to look at the advantages and disadvantages of each of the 3 approaches.

2.2 Packaged software

Advantages

Most of the software that an organisation uses will be off-the-shelf, and the same application will typically be run by thousands of other organisations. The advantages can appear compelling:

Cost savings – The most quoted advantage of the software package approach is reduced cost. This is because the cost of development is usually spread over the expected large number of users of the application. Whereas, with custom development the entire cost is borne by the organisation that commissions the application.

Time savings – With packaged software the application can be bought and used following installation and user training. There is no requirement for business process analysis, application design, programming and testing.

Feature rich – The software can be very sophisticated (e.g. Microsoft Word) as the forecast revenues from a very large numbers of users means that a lot of resources can be applied to its development and ongoing improvement.

Quality benefits – An off-the-shelf software application is usually a proven product that has undergone testing and user acceptance. As a result the application should be relatively error-free.

Available documentation – For an off-the-shelf application, the documentation can be inspected and evaluated before purchasing the product. This is usually of a high quality, as it is an important part of the selling process. In contrast, the documentation supporting a custom application development is often not available until very late in the lifecycle.

Available training – Prospective purchasers of an off-the-shelf application can attend a course prior to buying the product and so further evaluate the suitability of the package. Economies of scale allow the software manufacturers to provide high-quality training courses, supported by professional trainers, at a relatively inexpensive price.

Product maintenance and enhancement – Packaged applications are usually supported by a formal maintenance agreement that typically provides:

- Access to a help desk, where experts can sort out user problems
- Upgrades to the package that will correct known faults and also include new functionality defined and agreed with the user community in advance

The cost of this support and enhancement is spread across many users, and so can be provided relatively inexpensively to each individual customer. Such cost would be more expensive if it was borne completely by the organisation commissioning a custom development.

Try before you buy – The ability to examine the application in detail before purchasing it is clearly important. This is not possible in the custom approach to application development, where the product is not ready until the end of the project. Evaluation can also be helped by visiting reference sites, where the operation of the package can be observed and user comments and experiences documented.

Disadvantages

Over complexity – As packaged software is designed to meet the needs of businesses in general and not the needs of a specific company, this means that the 'feature-set' must work universally. Consequently the application is usually feature rich and will include large sections that most organisations will never use (e.g. the average Microsoft Word user is reckoned to use only about 10% of the available facilities). In effect you are paying for what you do not need.

Compromise – By its very nature, packaged software is designed for many different types of users, each of which will have different requirements. In certain circumstances this may mean that you have to alter the way that you work in order to fit in with the way that the software has been designed. While at the other extreme there will probably be functions that you require that are not included with the application.

Support issues – As one voice amongst many others, your requests will not carry much weight with a large software manufacturer. If you have problems you are usually at the mercy of a large and faceless organisation who may not be quite as concerned as you are if you have a major problem that needs to be fixed immediately.

No competitive advantage – As the same application can be bought by your competitors, it is very difficult to gain any competitive advantage from its use.

Ownership – With an off-the-shelf package the ownership of the software usually remains with the software manufacturer. Customers are licensed to use the product, but they never own it. This means that the software manufacturer can make decisions about the ownership and support of the product, such as deciding to withdraw support from earlier versions of the package.

Input to future development – Unless you are a very large user of the application (i.e. purchase thousands of licences) your future requirements will be of little interest to the software manufacturer. They may also take the decision not to upgrade the package, so you could end up with a legacy system that leaves you at a competitive disadvantage.

Legal redress – The legal responsibilities of an off-the-shelf software application provider are complex. The licensing agreement is defined in favour of the manufacturer with a clause that usually 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. Many customers do not take this responsibility seriously and are unable to properly assess whether a particular package supports their needs.

2.3 Custom software

Custom software development can come in 2 forms. Firstly, where you have an existing application you can tailor it to better match your actual needs and secondly you can develop an application from scratch to meet your precise needs.

2.3.1 Tailored software

Tailored software offers all the pros and cons of packaged software but with the major advantage that you can obtain a solution that is customised by third parties to match the way you work. This is usually at a fraction of the price of developing an application with same features from scratch.

While this may appear very attractive there is a down-side and that relates to support of the original package. As mentioned above, with packaged software you have no control over the future development and on-going support. If the manufacturer of your off-the-shelf package decides upon a significant upgrade that significantly changes the architecture of the end-product you may be effectively 'left-behind' unless you pay for an upgrade of the tailored part. In the worst case the customisation may need to be totally redone from scratch.

Similarly, if the software manufacturer decides to end support for an older version of the product – usually done to force users to upgrade to the latest version and boost their revenues – again you are out on a limb. It also means you may be forced to pay for more tailoring to ensure your business is not adversely impacted by the 'upgrade' you did not really want in the first place.

2.3.2 Bespoke software

Bespoke software applications are written from scratch to meet an organisation's precise and unique requirements. As a result of these they provide a number of specific advantages:

Advantages

Fits YOUR requirements – It has been specifically designed and tuned for your particular needs and can be implemented to fit exactly with the way that your organisation operates. In addition, as business stakeholders and users are involved in creating the optimal solution, it will provide you with performance benefits that are just not achievable with packaged software.

Works closely with your business processes – A bespoke solution can incorporate business processes that are specific to you and which do not exist in any packaged solution. This also means that it is more flexible than packaged software and can be modified and changed over time as your requirements and business practices change.

Integrates with other software – The bespoke application can be designed to interact with other software that you utilise and provides the potential for a fully integrated IT infrastructure across the whole of your organisation.

Easier to use – Users will find it easier and more intuitive to operate as it will not contain unnecessary or superfluous functionality and will function in the way that they would like it to work. This in turn will lead to reduced training requirements and associated costs.

Enhanced support – You will receive a much better level of support, and can in many cases talk directly to the analyst or architect of the solution. A good analyst or architect can also significantly add value to your company by suggesting alternatives, improvements and by acting as a source of IT advice and information. In addition you will be in control of future developments and will not suddenly find support being withdrawn in an effort to force you to upgrade.

Competitive edge – The use of a professionally developed bespoke software application can give you a significant business advantage over your competitors as it will simplify, optimise and streamline the processes that differentiate your business.

Disadvantages

Source code – If you do not take possession of the source code, when the application has been completed, you are dependent upon the developers continuing existence and goodwill.

There are a number of ways round this problem. Firstly you can make sure that your contract allows for the delivery of the source code to you when the application is implemented. Alternatively you can set up an Escrow agreement. A software Escrow agreement is the holding of source code and associated technical information and documentation by a trusted and independent third party. This ensures that in the event of supplier failure, you are able to maintain business continuity by getting hold of the code for your critical application.

Standards – If the software is not designed and developed properly it may be unstable, unreliable and full of bugs. Selecting a developer with a proven track record and appropriate accreditation will avoid this problem. In addition it is always advisable to take up references and talk to some of their recent clients.

Higher costs – The investment required will usually be much higher than for packaged applications. However the initial investment should not be viewed on its

own, but as part of a wider picture as greater productivity and efficiency gains will ameliorate the higher costs over time.

Many organisations undertake a business justification exercise and compare the initial costs against the expected benefits and commercial advantages. As a result of this the correct business decision will be made. In addition it is also possible to apportion the development into phases, thereby spreading the costs over time.

Advantages and disadvantages: summary

Packaged software		Custom software	
Advantages	Disadvantages	Advantages	Disadvantages
Cost savings	No competitive advantage	Competitive edge	Higher costs
Time savings	Compromise	Fits your requirements	Source code held by developer
Feature rich	Over complexity	Works closely with business processes	Potential QA issues
Quality	Ownership	Source code ownership	
Documentation	Future development	Integrates with other software	
Training	Support issues	Enhanced support	
Maintenance	Legal redress	Easier to use	
Evaluation	Potential QA issues		

3. Guide to developing successful custom applications

Successful bespoke applications do not happen by accident. To ensure success it is important that a staged approach is followed.

3.1 Business process analysis

Increasingly, organisations are realising that the first step in almost any application development project is to analyse and define their business processes. This is true whether the project is completely bespoke or tailoring an existing application.

Business processes represent a way of looking at an organisation and what it does, as opposed to the traditional departmental or functional view. A business process is a set of logically related business activities that combine to deliver something of value (e.g. products, goods, services or information) to a customer.

The analyst will work with the business owners and stakeholders (e.g. users) to map current processes, consult and detail critical business factors and gaps. Typically this will cover:

- The company's strategy
- Business value and benefits of the proposed software solution
- Economic, technical and operational feasibility
- Requirements gathering
- Stakeholder involvement
- Risk analysis

The report produced will clearly show the elements of the project that are business critical and those that will provide the highest return on investment. It will also enable the customer to identify the cost-benefit of the total project including all add-ons and configuration.

Business process analysis is critical to achieving a successful custom application and ultimately gaining consensus, buy-in and adoption from both management and users.

3.2 Functional and physical requirements

Once the business process analysis has been completed it is possible to start mapping out the functional and physical requirements of the application.

The requirements document should clearly state what the application must do and also show the other attributes it must possess, it should:

- Expand upon the initial requirements outlined in the business process analysis
- Define the planned features and functions of the application
- Describe other qualities that the application must have, such as usability attributes or regulatory compliance
- Clearly define and prioritise the customer requirements and software capabilities and features that must be delivered in the final product

The functional specification includes specific information about the requirements of the application, and for each should describe:

- Purpose - what the function is intended to accomplish
- Input - what inputs will be accepted and in what format, sources for the inputs, and other input characteristics
- Process - the steps to be performed, algorithms, formulas, or techniques to be used
- Output - desired outcomes such as the output form (e.g. report layout), the destination of the output, output volume and timing, error handling procedures, and units of measure
- Usability requirements - these are features that ensure the user friendliness of the software.

The overall aim being to produce a blueprint and visual specification that will enable stakeholders and users to fully understand the functionality and outputs of the proposed application.

In addition the following physical requirements need to be considered:

- Hardware requirements at both the server and desktop level
- Additional software requirements
- Network requirements
- Security

3.3 Development

Once the functional and physical requirements have been agreed and signed off by the customer, it is possible to begin work on the development and coding of the application.

There are 2 ways that software can be developed: phased or iterative.

3.3.1 Phased development

Phased development (also known as the waterfall approach) is based on a set of activities applied sequentially along the development life cycle. Each development stage is accomplished by testing of its work products, before proceeding to the next phase. When defects are revealed through testing, they are fed back to previous stages in order to be corrected.

The advantage of this approach is that it will produce an agreed specification at the outset, enabling time and costs to be accurately calculated and controlled. A potential problem with this approach is its strict policy of accomplishing a certain phase before proceeding to the next one.

3.3.2 The iterative approach

The major difference between the iterative and phased approaches is that the phases of the waterfall process may be applied iteratively throughout development, and are known as "workflows" rather than "phases". This method typically produces a better product than the phased method, but does have its disadvantages.

The iterative approach starts from the point of view of a loose specification that is firmed up in the cycles of design, implementation and client review until a working solution that the client is happy with is produced.

This provides a more flexible approach than phased development and enables features and functionality not considered at the outset of the project to be easily added. But the drawback is that costs and time may spiral out of control as more and more iterations are added to the development cycle.

3.4 Implementation and deployment

When the application, or key components of the application, has been developed and tested it can then be implemented. How this actually happens in practice will vary from organisation to organisation.

It is usually a good idea to implement the application in a pilot situation first. This will enable you to spot any potential problems while they are at a small scale, rather than attempting to roll-out across the entire user base in one go.

Any problems or issues identified at this stage can then be fixed before the rest of the organisation is affected. Other factors that need to be considered at this stage include:

- Transitioning from the old to the new application and processes – it is important to educate the users on the benefits and reasons for change to the new application. Many well designed and developed applications have failed purely due to user hostility
- Documentation – both at a user and technical level should be available when the system goes live
- Data Migration – planning for this should take place early on in any project to ensure that current data is immediately available for use in the new application
- Hardware readiness – servers, desktop PCs, printers and the network infrastructure should all be reviewed and upgraded where required, before the application goes live

3.5 Training

Training is a critical part of any custom application, as it is the end users who will ultimately deliver the business benefits envisaged when the application was first planned.

Most users do not like change. That is why it is important that training should start before the application is due to go live, so the users can actually “touch-and-feel” the application. At this stage it is all about educating them as to the benefits of the new application, and the reasons for change. This way, when they actually get to use the application for real, they will be looking forward to it, rather than viewing it with trepidation.

Training can take many forms, dependent on the best method and budget and can include:

- Formal classroom training (one to many)
- At their workplace (one to one)
- Computer-based or e-learning

Other factors to consider are having someone available to “floor-walk” on the day the application goes live. This can help to remove potential problems by solving them when and where they arise.

3.6 Support and maintenance

Once the application has been successfully implemented it is important that you receive ongoing support and maintenance of the application. This ensures that any bugs that do not immediately come to light can be fixed when they do.

In addition you will be able to make enhancements to the product and further fine-tune it to meet your adapting and changing needs. New functionality can also be brought in over time.

4. Conclusion

While there is no doubt that packaged software can be the right solution in certain cases e.g. word processing or spreadsheet production, its limitations can be a barrier to growth if it is used in other areas.

This is especially the case where you have business processes that are particular to your company and provide you with a competitive advantage. It is highly unlikely that any packaged software can provide optimal benefits without major customisation.

This is the area that custom applications come into their own. Whether you tailor an existing or off-the-shelf package to meet your needs, or design a bespoke application completely from scratch, the benefits can be enormous and far outweigh the initial costs.

Going forward, this approach also provides flexibility. As your business grows and requirements change you will be able to adapt the application to meet your changing needs – something not always possible with packaged software.

Finally, it will leverage your competitive edge, in providing you with a unique system that fits your precise needs and fully addresses your particular challenges and circumstances.

5. References

1. Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach, 2nd Edition. Alan Dennis, et al. 2005
2. Analyzing Requirements and Defining Microsoft .NET Solution Architectures. Microsoft Press, 2003

At Fletcher IT Solutions we deliver superior, process-oriented solutions that are highly usable, dependable and secure. We are able to do this because we combine strong business analysis skills with an in-depth knowledge and experience of the technology.



FLETCHER IT SOLUTIONS

Beechfield House
Lyme Green Business Park
Winterton Way
Macclesfield SK11 0LP
United Kingdom

Telephone: +44 (0)870 766 1824

Fax: +44 (0)870 766 4957

Email: enquiries@fletcherit.co.uk

Web: www.fletcherit.co.uk



Associate Member of the
Institution of Analysts
and Programmers



Microsoft Certified
Partner with .NET
connected certification