Sponsored by mobysoft

Should I Build or Should I Buy?

A consideration paper on the pros and cons of building in-house software vs. buying an independent software vendor's product.

Author : Susan Morrow Sponsor : Mobysoft

Should I Build or Should I Buy?

A consideration paper on the pros and cons of building in-house software vs. buying an independent software vendor's product.

Contents...

About The Author

The Author: Susan Morrow, Head of R&D at an independent software vendor, has worked in the IT industry, in particular in software development, since the early 1990s.

In that time she has worked on projects ranging from small, localized installations, to extremely large, multi-consortia, multi-national software implementations.

She has found herself on both sides of the build vs. buy fence, working within an ISV organization running a development team, as well as hiring and managing third party development companies to build product.

This has given her a well grounded and practical insight into the software development process and all of the trials and tribulations it can bring.

1.0 Build it or Buy it?

Business processes are becoming ever more digitised helping us to solve specific business problems. When we come to choosing the software to solve that problem we often find ourselves asking the question, should we 'build it', rather than buying a commercial off-the shelf (COTS) solution. This 'build or buy' question is one that companies have had to grapple with since the advent of computing and with the fast pace of business digitization this decision is becoming ever more important to get right. There is no straightforward or generic answer to the build vs. buy question. Instead, it is something that needs to be dealt with on a case-by-case basis. To make the correct decision it requires a deep understanding of the areas impacted by that final choice. Buy or build comes down to some fundamental factors such as timing, costs and available resources. It also is dependent on other criteria such as the type of software; some software requires highly specialized knowledge to build, or requires deep knowledge of web security, for example. The open source community can offer a potential starting block for build it yourself projects, but even open source software can require specialist knowledge to productize.

Occasionally, the decision to build or buy is made for us, for example, when there is simply nothing on the market that covers the requirements of the project. The decision is then simple; if we absolutely need this solution then we have to build it ourselves. Building a product from scratch involves many moving parts across the development lifecycle. Understanding how to recognize if building it yourself will become a decision you wish you'd never made, must be based on experience and knowledge of the development process. Coming at the choice from an informed position helps to make the build vs. buy decision easier and with less risk.

2.0 Build it and they will come...

I've been working in software development for over 20 years, running development teams, working on very large national projects and handling multi-consortia development programs. One thing I have learnt in that time is that software development is a process made up of many interconnected moving parts, each inter-dependent. Software projects are notoriously prone to failure too. The Standish Chaos Report[1], which in 2015 looked at 50,000 software projects across the globe, found that 19% of them failed, 52% were 'challenged' and 29% were seen as a complete success. Building software is not for the faint hearted, but it can come with its rewards, including having full ownership of the product intellectual property (IP) and full control over future enhancements.

Of course, software comes in all shapes and sizes. A simple application that performs a one off accounting task is a very different beast to one that is web-connected, manages user accounts and performs complex tasks based on specialist algorithms such as encryption. Making the decision to build your own application really does depend on what that application needs to do and if you have the right resources in place to build it. In any case of build your own, there are certain key considerations to add into the decision making process, these include:

2.1 Developing Your Requirements

Mapping out the requirements of the system is the first step in creating your build. Requirements will dictate the ultimate features and functionality of your application. The requirements specification itemizes both the functional and non-functional needs of the solution.

Functional aspects of a software application include items that represent what the software does, for example, specific features of the product, audit capability, administration, authentication, user account variables, interfaces and so on. Non-functional variables include items such as security, data integrity, environment, servicing of the system and disaster recovery options.

This is arguably the single most important step in the creation of a software application. If it is done well, it can save much time in future re-design or re-specification. Well thought through requirements can avoid project slippage and reduce costs. Even with an agile approach to software design, robust, well considered, requirements, which carefully set-out non-functional issues, such as, interoperability with existing systems, scalability and maintainability, will give a project a greater likelihood of success.

As you go through the requirements analysis of the product, you may well find that this activity focuses attention on the potential skills gap you may have in-house. The requirements should identify areas where you do need specialist knowledge to complete the build so that it is robust and fit for purpose. A comprehensive requirements specification will also give you a more precise view of the type of resourcing you will need to take the project through to completion.

[1]Standish Chaos Report 2015: https://www.standishgroup.com

2.2 Finding the Right Team

Let's suppose you are building an enterprise software platform that is complex and requires knowledge of particular protocols, security or algorithmic manipulations. Or let's say you're building a Cloud based web application, where functionality and protocols are changing at a fast pace. Before you even start, you need to know if you have the right people in your organization to handle this type of project and who have the skills to keep pace with changing technical remits. Even if you decide to outsource the development to a third party (on, or off, shore) you will need people who know how to manage and direct third party developers and who can 'talk the same language' that they do.

Development team members need to cover the full remit of software design, creation, testing, implementation and support.

The development process requires that you take your requirements specification and from this you generate documentation that describes the design specifications, user journeys, and so on. The system and security architects document the architecture of the software. This set of schemas and documents can then be used to determine development tasks and allow the development team to begin work on the solution.

Team members come from multiple disciplines; this can depend on the type of product being developed. For example, it may require input from stakeholders with specialist knowledge, such as financial or legal expertise, acting in an advisory position. As well as software developers, the process also requires business analysts, designers, user interface experts, software testers and implementation specialists. Depending on the product and the industry, there may well be a need for specialists to carry out checks to make sure the software is compliant with regulations such as ISO 27001.

Type of Skills Needed To Build It Yourself

Team members need to come with a multitude of skills and experience. The software development lifecycle requires knowledge across a number of areas:

- **Project management**
- **Requirements analysis**
- **User journey development and related documentation such as state models**
- **Architecture design, including the review of certain protocols, database types, security measures, etc.**
- **Storyboarding and user interface design**
- **Management of usability / accessibility testing**
- **API development where needed**
- **Technical authors for user guides and other documentation**
- **Test case and automated script creation**
- **Testing of software across CIT/SIT/UAT and pre-production**
- **Environment design and setup**
- **Implementation**
- **Maintenance**

The team you pull together will need to know and understand the complexities of software development and delivery. Companies like Facebook regularly create their own in-house software builds for internal use. However, this is their core business; they already have thousands of developers. In an industry where this isn't part of the core business, in-house software development can be a challenge.

2.0 Build it and they will come...

On-going Challenges: Recruitment, Knowledge Transfer and Retention

The challenges of creating that 'dream team' go far and beyond knowing whom you need. Recruitment especially of specialists can take time. If you are looking for developers who have knowledge of highly sort after skills, then you will be in competition on at least a national scale, perhaps even global, as more companies offer remote working capability for development staff. Top performing software designers, developers and architects are a premium commodity, with over 100,000 jobs advertised in the IT sector in February 2015[2]. A survey by recruitment agency, Robert Walters[3], found that 72% of businesses are struggling to find talent.

Even if you pull your team together, you then have the issue of retention. Software developers especially like to build. Once they've built your product, if you don't give them new, exciting projects to work on, they tend to move on. Tenure of software developers and other IT workers is renowned for being short. There are actually a number of reasons for this, but the fact is that, on average, in the UK, IT employees stick around for no longer than 2 years[4].

This fluid situation creates further issues around knowledge transfer. One of the greatest hurdles in ensuring a seamless continuation of work is in the knowledge transfer cycle. Moving information from one employee to another, or even an employee to department and vice versa is difficult and error prone. It can take a lot of time to get new developers up to speed with existing code bases. This becomes quite labour intensive when new developers come in to maintain existing code bases.

2.3 Supporting Your Application

Keeping the software current

Once you are into the production phase, this is not the end of the project; it is the beginning of the next phase. Support of your software isn't just about technical support of users. It is also about maintenance and the updating of the software. As we've seen with the advent of Cloud computing, many software applications are migrating to a Software as a Service (SaaS) model, this has required older desktop architectures to be is re-engineered to work in a Cloud environment. For a company that has already invested large resources into a product, paradigm shifts like the Cloud, or the advent of mobile apps, can result in going back to the drawing board, or worse, back to an independent software vendor.

These types of worst-case scenarios do have to be figured into the equation when deciding to build your own application. On-going upkeep of a software product, in an ever-changing technology environment is a key consideration in the build vs. buy decision.

User support

Support of your user base is a key to the seamless uptake and acceptance of your application across your company. You should have some form of support available, for example in-app chat, ticketing system, or help desk availability.

[3]Meridian Recruitment: https://www.meridianbs.co.uk/blog/2016/01/whats-the-most-in-demand-job-for-2016

[3]Robert Walters, Recruiting Professionals in a Candidate Short Market: https://www.robertwalters.co.uk/content/dam/robert-walters/country/united-kingdom/files/whitepapers/recruiting-professionals-in-a-candidate-short-market.pdf

[4]Auriga, Employee Tenure becomes Hot Topic for tech Companies: http://auriga.com/blog/employee-tenure-becomes-hot-topic-for-tech-companies/

2.4 Hidden Time / Costs

Because hidden costs are, hidden, they usually get missed off ROI calculations and so can mean the make or break of in in-house development project. One particular analysis of this phenomenon was done by PWC who looked at the impact of building and maintaining an in-house system, in this case for HR and payroll. The report highlights that 50% of the costs of in-house IT projects were hidden costs.

Hidden costs come in all areas of the development cycle. For example, development costs aren't just the salary of the developers you employ; you also need to service those people with the tools of the trade - we're not talking pizza and cola here, we're talking high specification computers, software development tool licenses, licenses for code repositories, code analysis costs, and so on. The peripheries needed to manage a development team must be taking into account.

Software testing should not be underestimated. The test process for any enterprise level software application needs to be rigorous to create a product fit for purpose. Testing is one of the areas that can negatively impact your Return on Investment (ROI) as defects are identified and sent back for re-work. And rework costs. This is especially true if any part of your development work is outsourced (see "Outsourcing obstacles" later). Industry analysts, Voke[6], have described enterprises as having a "cavalier stance" in managing the testing and defect/rework cycle, which ends up adding substantial hidden costs. Outsourcing can also have hidden costs, for example, travelling to meetings if you outsource the development work.

2.5. Outsourcing Obstacles

It may be that you decide to outsource some or all of the development work to a third party company. You may even have more than one company involved in the outsourcing. For example, you may want a development company that specialises in web and front end development and another that can work on the back end, core architecture of the application. Outsourcing, can be a great help in developing in-house applications. Outsourcing can allow you to find the right specialisms for your product development. For example, you may need developers who have specialist encryption knowledge, or who can build mobile apps, or who understand the use of certain Internet protocols and so on. Outsourcing can be a great way to supplement knowledge gaps in your in-house development team.

I have personally used outsourcing to augment my own development team, to free up internal resources to work on other projects and to manage a skills gap in the team. I was able to find an excellent, highly responsive, very talented third party web Development Company, who helped my team build the front end for a web application we were working on. However, there were certain hurdles that had to be closely managed, these included:

1. **Direction.** My team had to closely direct the activities of the outsourced company. Even though the company was highly proficient, they still had to understand the full remit of the design and integration of the front and back end through an API. Our in-house team had to create extensive and detailed documentation to make this hurdle easier to jump.

2. **Costs and change control.** The project was one where usability studies and accessibility tests were being carried out on a fairly regular basis. This meant that interface changes were frequent in the early stages of the project. I had to ensure that these changes were covered in the original cost plan. If not, the costs would have spiralled.

3. **Responsiveness.** One of the biggest issues with outsourcing is sharing tenancy. Third party companies, rarely have one customer on their books. They have to manage their customers and schedule in work for each. If, as I found myself on more than one occasion, you need an urgent fix for defect, this type of setup can cause issues and delays.

[5]Price Waterhouse Coopers, The Hidden Reality of Payroll and HR Administration Costs: https://www.adp.com/~/media/Solution%20Builder/Documents/PWC%20TCO%20Study%202011.ashx

[6]Voke, Do You Know Your Cost of Rework? Voke Publishes New Research on Reducing the Cost of Rework: http://voke.blogspot.co.uk/2014/09/do-you-know-your-cost-of-rework-voke.html

2.6 Exit Strategy – Managing Your Black Swan

IT projects are renowned for failing. There are many reasons why this is so, including, uncontrolled requirement changes (especially later on in the project), staff issues, spiralling costs, lack of specialist knowledge and technology changes during the project, as well as mismanagement of consultants and third parties. Analysts, McKinsey found that 45% of IT projects go over budget and 17% fail so badly that they threaten the company.

Of course you should plan not to fail. You can certainly minimize the likelihood of failure, but unforeseen events can make this a difficult task. Disaster recovery plans should be built to accommodate worse case scenario events.

If a build project fails, your organization needs to have measures in place to minimize the impact. Exit strategies should be planned up front as part of the project management. The more parties involved in the development of the application, the more complex the exit strategy will be. Exit clauses in contractors and third party development company contracts need to be included in this strategy to minimise financial impact.

Keeping stakeholders aware of project risks, whilst ensuring their buy-in, can also help reduce the pain of a forced exit.

[7]McKinsey and Company: Delivering large-scale IT projects on time, on budget, and on value:
http://www.mckinsey.com/insights/business_technology/delivering_large-scale_it_projects_on_time_on_budget_and_on_value

3.0 Buy It In – Reinventing the Wheel is For Others

If you decide that building your own product is just too much of a tough ride, then the alternative is to source the technology from an independent software vendor or ISV. The likelihood of there being a product in the market place that offers a solution to your needs is high; after all, software vendors create products to sell, which means they build to known market needs. In addition, an ISV will be an expert in their field. The vast majority of ISV's are in heavy competition with other software vendors and need to keep, not only pace with the market space, but also outpace it with new innovations. They also need to offer extremely high levels of customer care to stay competitive.

Having a close relationship with a software vendor can, in some instances, also mean that you can have some input into the future development of their product, especially if that feature is commercially viable.

Having off-the-shelf software that can be up and running quickly may mean the difference between your company being competitive or not. Of course, buying ready built software isn't always a case of 'plug and play', it may involveconfiguration and hosting decisions and setup, but in general, an off-the-shelf product will be in situ more quickly than an in-house built one.

What you generally don't get from using bought in products is the ownership of the intellectual property (IP) of the product. The software vendor retains that, even if you have input into new features.

3.1 Software Vendors are Experts in their Field, Right?

To design and build a software product and release it into a highly competitive and often global market, you really need to know your stuff. If you don't your product will simply fail and be lost in the sea of other similar products.

To create a software product, especially an enterprise type software platform, you need to have in-depth market knowledge, often gleaned from working in the industry you are serving. Software product ideas often come from individuals who have faced similar needs and issues to you. They have seen an opportunity and decided to create a software house to resolve that issue. You often see very large software vendors, like IBM or Computer Associates buy up small software companies; they do this in many cases to buy in expertise in specific areas, rather than trying to build the product directly.

Independent software vendors have to employ specialist software designers and developers who understand the specifics of the area they work in. For example, any web-facing application, or one that has user accounts, will be open to the myriad of cyber security threats that exist. Security of software applications is now a fundamental part of the development process. A software architect working on this type of software must have a deep understanding of the types of threats the software will need to be hardened against. The developers need to know how to create secure code. The software testers need to understand how to PEN test their product internally before it is released onto the world stage. Software platforms that perform specific tasks, for example financial platforms, often have complex underlying algorithms that need to be highly accurate, fast and built by knowledgeable persons. The use of industry specialists to create world-class software products is a basic requirement of ISV's to allow them to compete.

But even the most competitive product is nothing without excellent customer support. ISV's need to compete in this area too, this often includes being able to offer bespoke solutions, based on their core product, help desk and administration options and first line 24/7/365 support to their customers.

3.2 Finding the Right Product at the Right Price

When setting out to buy in a software product, you will benefit from doing market research and due diligence. As mentioned previously, in most areas of software there will be more than one product available. It is important before you start to map out the requirements you need to fit the remit of your project. This can form the basis of your competitive analysis of the products available.

A competitive analysis should give you sight of the products capability, but due diligence should also extend to include the company itself. The case history of the failed NHS patient record system[8], which ended up costing the UK tax payer around £6-10 billion, is an interesting case of poor due diligence leading to project failure. It is as important to choose the right company, as it is to choose the right product. The company attitude and ethos should match your expectations.

3.3 Do the Product Features Fit Your Remit?

Once you've determined a short list of potential products that fit your requirements you should assess if the products give you all of the features that you need – if not does the company offer some bespoke options? Off the shelf, doesn't always meet the requirements of an organization. However, bespoke work could be minimal, as long as the core platform meets most of your requirements. Software that is well designed is built to be extensible to allow additional features.

3.4 Vendor Dependencies: Support and Hosting Options

Building a relationship with your ISV can be highly beneficial. The ISV becomes more aware of your needs and can tailor their support packages to those needs. Software is ever evolving. This is true if you build it yourself or buy it in. ISV's will often help you with additional platform requirements such as hosting. They are experts in their product, so they will be able to help optimise environments for performance and cost.

Software updates are something to consider too. Vendors will have an update schedule that outlines expected software releases. This is helpful in allowing you to plan out human resource needs and build in any contingency planning for offline and maintenance time.

One of the potential downsides of using an off-the-shelf product is the dependency aspect. There is always a worry with smaller vendors that they may be bought out by a larger vendor, who may be less inclined to give you personalized support or who may go into liquidation. The former issue can be resolved through contract clauses to ensure you get continued support at the same level. There is potential in some circumstances to use escrow agreements where source code is lodged with a third party, accessible if the worst happens, but this isn't always the best way to deal with this.

In larger companies, product deprecation is something that could potentially happen (as anyone who still uses Microsoft Windows XP knows). Again, contractual obligations can be set that can mitigate the impact of this.

[8]MThe National Program for IT in the NHS: A Case History, Campion-Awwad. O, et.al. Cambridge University, February 2014: https://www.cl.cam.ac.uk/~rja14/Papers/npfit-mpp-2014-case-history.pdf

3.5 Exit strategy – Managing Your Black Swan

If you are running a large IT project, this might overlap with several software and hardware vendors. Running complex IT projects needs flexible and responsive project management to keep it on track.

One way to help mitigate and minimize the likelihood of outright failure is to add penalty clauses to the contracts of the companies you are dealing with. There is nothing that focuses the mind of a company as much as the financial bottom line.

If the project continues to fail, you need to ensure that your organization can manage the impact. Have a back up plan in place for all of the possible failure points in the project, ensuring worst case scenarios are thought through can bring a lot of creative thinking into the risk management process.

4.0 Build vs. Buy: The Big Question – ROI

Return on Investment or ROI is your starting point in the build vs. buy decision making process. Calculating your ROI and understanding the financial variables of each option is a great place to work from before you can make an informed decision about which route to take.

Probably the hardest place to start is to itemise where the costs come in. In a build scenario, there are often hidden costs (see the section 2.4). Also in a build scenario you may have to estimate costs of bringing in specialist staff or using outside contractors.

With buy in options you should be able to get accurate quotes that cover all of the costs from the vendor. Make sure they do cover all of the costs, including maintenance and support and any costs associated with upgrades.

Build it:

You should expect to include the following costs in a build it yourself scenario:

1. Research of requirements
2. Project management costs
3. Cost of specialist design team and business analysis consultants
4. Costs of internal development team
5. Costs of outsourcing, if using
6. License costs for hardware and software development tools
7. License costs for associated software such as databases, digital certificates, etc.
8. Costs in time to market
9. Test support costs, including UX /accessibility testing, setting up and costs of test servers
10. Technical authors for user guides and other documentation
11. End user training
12. Environment costs
13. Code analysis costs (if performing)
14. Maintenance and new feature deployment (post production)

4.0 Build vs. Buy: The Big Question – ROI

Buy it in:

1. Software license costs of the product
2. Renewal costs over period of contract
3. Support and maintenance contract costs
4. License costs of associated products, e.g. database license, digital certificate costs, etc.
5. Project management costs
6. Implementation costs
7. Escrow costs (if using)
8. Consultancy and advise (if using)
9. End user training

5.0 Roundup

In the cases of build vs. buy, whichever decision you make will be a unique one and one specific to your organization at a given time and place.

However, there are a number of critical considerations that can sway your decision towards choosing to buy, rather than build:

1. **Time to market:** if you choose to build over buy, the technology landscape may have shifted massively underneath you by the date of delivery. This can result in at best, major adjustments to project execution, and at worst, a product not fit for purpose, which can materially impact your company's future. We've seen this technology shift happen in recent years with the advent of mobile, Cloud and more lately the Internet of Things (IoT) paradigms. A bought in software application, from a dedicated software vendor, is much more likely to have kept pace with technological advances, simply because the vendor will have to stay competitive within their space.

2. **Resourcing issues:** Endeavours to find the right people may cause delays in project start date, and can even mean a stalled project if you cannot find the right level of skills. This is more pertinent than ever in an increasingly competitive market for experienced IT professionals. ISV organizations are built around having the best and most skilled staff in a given specialism – if the ISV does not have excellent staff, they will not have been able to create a competitive product in the first place. The best ISV's are the best in their field.

3. **Hidden costs can cause projects to fail:** These costs can slip into the project operations and funding them not only takes enormous amounts of time to organize, but can cause major project issues, including project failure. Hidden costs are one of the most prevalent reasons for IT project failure.

Making the final decision is one that ultimately has many considerations, some of them very specific to your organization. This isn't just about the financial costs of the project, the time to market, or the availability of the right people. It is also about the appetite that your company has for taking on, what may turn out to be an epic project with many twists and turns, as you attempt to build a complex piece of software and maintain it going forward. The decision to build software should never be taken lightly, it can have ramifications across the organization as you attempt to bring the right team together and configure the necessary designs. It may be that you feel you are in the right place to embark on creating your own in-house solution, but this should be done with eyes wide open. If a solution already exists in the marketplace, you can be sure that it has not been done overnight. The ISV that has built the solution will have gone through the painful hurdles that you would need to go through if you choose to build over buy. The most fundamental question you should be asking is, do I really want to take on such a commitment, only to reinvent the wheel?

Sponsored by

mobysoft

0161 974 2000

www.mobysoft.com