WHITEPAPER

Software Buyer’s Guide: Build vs. Buy

Author: Colin Hung, P.Eng.

Debunking 6 arguments that you *must* ignore when deciding whether to build or buy software

software for
safer healthcare

rl solutions™

Summary

Over the past few years, the options available to organizations looking to implement technology-based solutions have increased. This is due to the rapid evolution of software development techniques, the wealth of software authoring tools, the move towards open-source software and the accessibility to quality software packages from vendors around the world. However, this increase in choice has not helped answer one question that continues to challenge organizations of all sizes: is it better to build a custom software solution or buy a commercial-off-the-shelf (COTS) package from a software vendor?

Given the constant state of change in the software industry, it is surprising that pundits on both sides of the build vs. buy debate are still using the same six arguments. Building vs. buying is a discussion that every well-functioning organization should have when it comes to software. They key to making the right decision for your organization is to consider relevant arguments on both sides of the debate and discount the irrelevant points. This whitepaper will summarize the six arguments that you should ignore.

Who should read this whitepaper?

RL Solutions wrote this paper with the intent to provide organizational leaders and internal project champions with sound advice for navigating the ‘build vs. buy’ dilemma. This paper is appropriate for non-technical & technical individuals at organizations that are thinking about, have recently started or are in the midst of an initiative involving software.

Introduction

For as long as commercial-off-the-shelf (COTS) software has been available, organizations in all industries and of all sizes have been faced with a challenging choice: build software solutions using in-house resources or buy market ready, innovative products from leading software vendors.

People have put forward many arguments that favor one approach over the other – some have even become generally accepted as truths. It is clear that many of these traditional arguments are no longer valid given the current state of software maturity, technology advancement and worldwide competition.

This paper debunks six such arguments in the build vs. buy debate – three in favor of building and three for buying. This paper encourages you to ignore the following 6 arguments in your discussions and focus instead on other more relevant points.

BUY arguments that you should ignore

- Buying is better because developing software is not a core competency of your organization.
- Buying software from a vendor provides a lower total cost of ownership.
- Buying software is less risky because you’re relying on a few key employees if you build your own solution.

BUILD arguments that you should ignore

- Building is a better option because you’re not locked in with a particular vendor.
- No products are available that match your needs, so building is better.
- Vendors always overstate their product capabilities & delay new features, so go with in-house resources that you trust.

BUY MYTH #1: Building software is not your organization's core competency

'Buy' pundits often say that organizations should stick to their core competencies and only focus on those activities that their clients value if they want to thrive. Since building software applications is usually periphery to client success, buy pundits argue that building software solutions should not be a core focus of most organizations. In other words, organizations should stick to what they do best and opt to buy COTS software rather than build their own solution.

This is an interesting argument, but one that falls apart after closer inspection. When you think about it, the only type of organization where building software is a core competency is a software vendor. And who are the most ardent supporters of the buy option? Software vendors.

Even if you accept this argument at face value, it does not hold any weight when you consider how ingrained computers and software have become in our business and personal lives. In the pioneering days of computers (1970s through the early '90s), this buy argument was accurate since the proliferation of computers and software technology was limited. There were only a handful of individuals with the skills and experience to run those computers – and they commanded a premium. It was better for most organizations to buy COTS software since software vendors could build better solutions with their specialized and more plentiful resources.

However, since the mid-1990s, a confluence of industry factors has lead to the rapid maturity of computers and software technology:

- The near-ubiquity of computers at work and at home has significantly increased the level of computer literacy
- The worldwide talent pool of computer engineers and software developers has exploded
- The dependency of daily work activities on software
- The easy access to advanced software development tools
- The sharing and implementation of software best practices in IT departments

The net result is that internal IT resources can now rival software vendors in terms of talent, structure, process and tools. Indeed, in the past few years, several software solutions developed by in-house resources have spun off into separate software entities. For example, Credit Suisse's spinoff of its application to manage virtual resources into DynamicOps and Wake Forest University Baptist Medical Center's spinoff of MRI Cardiac Services Inc.

Hospitals, consulting firms, financial institutions, manufacturers and other are perfectly capable of acquiring the necessary skills and infrastructure to build quality software solutions. Thus, you should no longer factor in the argument that in-house resources cannot develop software as well as software vendors into your build vs. buy debate.

BUY MYTH #2: Buying provides a lower total cost of ownership

In the past, the cost of building software in-house was disproportionately higher than buying similar COTS software. This was because programming languages were extremely specialized and the time & expense associated with acquiring those skills was disproportionately high. Illustrating this gap was easy using an example similar to the following:

COTS software

Annual Costs	Cost Description	Estimate
Year 0	Purchase cost of software	$50,000
Year 1	Maintenance + support	$8,000
Year 2	Maintenance + support	$8,000

Equivalent in-house software

Annual Costs	Cost Description	Estimate (assuming $150/hr)
Year 0	Development labor @ 10,000 hours	$150,000
Year 1	Maintenance + support @ 100 hours	$15,000
Year 2	Maintenance + support @ 100 hours	$15,000

Based on the above analysis, it is far more cost effective to buy a COTS software package since it is 60% less expensive after two years of ownership ($66K compared to $180K).

While these numbers are impressive, they are based upon an assumption that the in-house solution needs to match the COTS package exactly in scope and feature set. This assumption ignores reality: where does it say that an in-house solution has to match a COTS package feature-for-feature?

When organizations undertake a project, the first step is (or, at least, should be) defining the project's scope and operational requirements. Often the requirements are a subset of the functionality that is available from a COTS package. A more appropriate comparison would be to take the cost of the COTS package and compare it to an estimate for your in-house team to build a solution that matches your real needs. In fact, from the author's experience, both options tend to end up at very similar number.

Furthermore, if your in-house technical resources do not charge their time back to the requesting departments, then it is debatable whether the internal 'cost' is really a cost at all. Certainly there is an opportunity cost to your organization, but there would be no actual cash spent if you consider your technical resource labor as 'sunk' costs. Given this reality, you should ignore the argument that COTS packages always offer lower total cost of ownership.

BUY MYTH #3: Relying on just a few internal resources is risky

This argument is based completely on fear – fear that the key personnel working on a solution built in-house might leave the organization, leaving the future of the in-house solution at risk. Imagine what would happen if your key developer left your organization in the middle of the project? Or, even worse, just as the project is set to go live – how would you recover?

Although this is a possible risk for some organizations, for the vast majority this is just an everyday risk of running a business. There will always be employee turnover, regardless of whether you choose to build or buy software. What if your key project champion left before you fully implemented the COTS package? What if your project manager takes ill right before training is about to start?

Organizations today are much savvier than 10 years ago and these types of resource risks are substantially diminished. Rarely does the success of a project rely solely on one person's shoulders. Instead, teams of skilled individuals are now the norm. These teams can adapt and absorb the loss of an individual due to improved work balancing and more disciplined project practices.

Furthermore, many IT departments have adopted technologies and best practices specifically designed to mitigate the risk of key employee turnover. For example, proper design and in-code documentation so that others can read what the components are supposed to do (even if they didn't create the components) and resume development work with little disruption.

As well, many of the current software development tools like Microsoft Visual Studio® include sophisticated code repositories that put all of the vital source code in one place (instead of scattered on individual workstations). Many IT departments have also adopted coding best practices including: code standardization, standard technology architectures, SOA, and modular design, which makes it easier for new staff to become productive (thus mitigating the risk of employee turnover).

Finally, this argument conveniently ignores the issue of employee turnover within the vendor's organization. In fact, you could consider the risk of employee turnover as a 'constant', meaning that it is the same for in-house and vendor development teams. Who is to say the vendor's own employees are less likely to leave the organization than your own? And if one of those employees was a critical contributor, that your organization would suffer just as much (if the vendor did not take the same steps to mitigate that risk as your own organization)?

So while the loss of key resources is something to consider for your project's overall risk, it should not a determining factor in the decision whether to build or buy a software solution.

BUILD MYTH #1: Don't get locked into a vendor's upgrade path

Proponents of the build option often use this argument to justify their position. They say that you are at the vendor's mercy in terms of upgrades and enhancements when you buy a COTS package. You will be just one voice among many and will be beholden to the will and whim of the vendor. Building a solution with in-house resources means working with people who are 'on your side' and will be more responsive to your needs.

This popular argument is laughably absurd for two reasons. First, lock-in is an issue no matter which option you choose: build or buy. As soon as you make a software choice, your organization is now locked into the upgrade and enhancement path of whoever is building (or has built) the solution. Whether that is your own internal IT team or a vendor, from an end-user perspective the lock-in is the same. You are now at the mercy of people who built the solution.

Second, this argument does not take into account the reality of the current hyper-competitive software market. With global competition and a wealth of low-cost options, software vendors have to be much more responsive to customers than they have in the past. Those vendors that ignore customer feedback and simply march to their own tune will not survive.

Perhaps this was a compelling argument in the past, when software companies could command customer loyalty, but in today's competitive environment, it is easy for an organization to switch to a new product if they are not satisfied with a particular vendor. Software vendors today offer a much higher level of customer involvement in the product design and enhancement process. This means that users are not 'locked-in' but rather 'included into' product development paths. For these reasons, you should ignore this argument against vendor lock-in.

BUILD MYTH #2: There are no suitable products

Individuals who favor the build option frequently bring up this argument to justify their position. They argue that because COTS packages can never match the end users' needs exactly, the build option is best. Their argument is based on their strong desire to avoid paying for features that they do not believe are needed or wanted. In their minds, going with a solution built in-house avoids excess features.

This argument is very tired and you should ignore it completely. First, it is very hard to believe there are no COTS packages that suit an organization's current needs. To put it bluntly, this argument may be a convenient excuse to end a search for a COTS package prematurely.

Second, 'fat' in a product is not necessarily a bad thing. Sometimes software has features and functions that an organization does not need today, but may become vital down the road. By encouraging you to eliminate excess features, build pundits who use this argument may actually be limiting the useful life of the software solution they create.

It's true that some features in COTS packages are unnecessary and perhaps even useless. However, it is also true that a COTS package may have a positive impact on the organization through one of those extra features. The most important reason why your organization should ignore this argument is the implication that settling for anything less than a perfect match between your needs and a software solution is unsatisfactory. While this is a laudable goal, it could leave an organization in a perpetual search for a non-existent solution.

BUILD MYTH #3: Vendors overstate their capabilities: Go with someone you trust, rather than waiting for a vendor

This is an interesting argument because it includes two different ideas. First, that you should treat vendors with a healthy dose of skepticism – especially when it comes to their stated product functionality and capabilities. Second, that waiting for your in-house team to build a system is better than waiting for a vendor.

While some vendors may overstate their product's capabilities (whether intentionally or unintentionally), this is certainly not true of all vendors. This is a gross misrepresentation of software vendors and the methods they use to get prospective customers interested in their products. It is also unfair to the many ethical and honorable vendors that are in the software business.

If you ignore the stereotype for a moment and consider the logic behind first idea, it still does not hold up. The argument implies that vendors overstate their capabilities, but in-house teams do not – not true. Although internal technology teams are not driven by profit or market share, they still have much in common with vendors. Both are driven by the need to impress end users with their technical skills, responsiveness, knowledge of the end users' environment and ability to deliver a successful project.

In short, in the build versus buy decision, the same goal motivates both vendors and in-house teams: to earn their customers' trust. Both vendors and in-house teams are therefore just as likely (or unlikely) to overstate their capabilities.

The second idea, that somehow waiting for a vendor is worse than waiting for internal resources, is hollow. It is safe to say that end users prefer not to wait at all – for anyone. While in-house resources are on the same team as their end-user colleagues, it's not logical to suggest that they should be trusted more simply because they work for the same organization. In fact, you can argue that it is worse for an internal development team to miss delivery dates because it pits coworkers against one another and can create unnecessary rifts within the organization.

As well, the wait for an internal team may not be any shorter than waiting for a vendor. In fact, you should carefully consider the starting point of your in-house team. If they are close to the 'zero state' – meaning that they have very little infrastructure (hardware and software) available, then the wait could be just as long as waiting for a vendor to enhance its system. Vendors are usually starting from a much stronger position since they already have an existing, fully functional system to build upon.

Finally, depending on the starting point of your internal team, are they not making the same timeline and functionality promises that a vendor is? For these reasons, disregard this argument in your build vs. buy debate.

Conclusion

The build vs. buy debate is challenging. There are strong opinions on both sides of the debate and the pressure to make the right decision is tremendous. Despite these challenges, having the debate is a sign of a healthy organization because it means that your organization is forward thinking and open to different approaches to solving problems. Organizations that squash this debate are stifling their users and suffocating innovation – not positive leading indictors of success.

The key to resolving the build vs. buy debate is to ensure that you only consider valid arguments and ignore the ones that are simply rhetoric. The six arguments presented in this paper have had their time in the sun and they have been thoroughly debunked.

This is not to say that you should ignore these issues entirely. When deciding whether to build or buy software, the issues of software as a core competency, the total cost of ownership, reliance on key individuals, vendor lock-in, lack of product choice and the gap between promises and reality are all worth considering. Just make sure that when you consider the issues, ask yourself first if the information is biased for build vs. buy.

About the Author

Colin Hung (B. Ac., P.Eng.) is Vice President of Alliances and Marketing at RL Solutions. He is responsible for creating an ecosystem of partners, leading the company's marketing efforts and setting RL Solutions' product strategy.

Colin is a 15-year veteran of the IT and software industries. Prior to joining RL Solutions, he was a Senior Director at performancesoft (now a division of Actuate) where he specialized in the development and delivery of Balanced Scorecard and performance management solutions. Colin has also worked at an IBM business partner, Ford Motor Company and the University of Waterloo in various IT and consulting roles.

Colin has a keen interest in how companies can apply technology to improve patient safety & quality. He has spoken at a number of conferences including the Crittenden Medical Insurance Conference and various regional SCHA events. Colin has contributed to the recent Taxonomy Monograph as part of ASHRM's Data for Safety Taxonomy Task Force.

Mr. Hung's credentials include a Bachelors Degree in Applied Science from the University of Waterloo where he majored in Mechanical Engineering and minored in Management Science. He is a registered Professional Engineer of Ontario.

RL Solutions designs cutting-edge healthcare software for patient feedback, incident reporting & risk management, infection surveillance and claims management. At RL Solutions, nurturing long-lasting relationships with our clients is what we do best. We have over 600 clients, including healthcare networks, hospitals, long-term care facilities and more. RL Solutions is a global company with offices in Canada, the United States, Australia & the UK.

For more information,
contact RL Solutions
Phone: 1 888 737 7444
Email: pr@rl-solutions.com
Web: www.rL-Solutions.com