

Buy or Build

Corporate Software Dilemma



By Lubo Zizakovic, President
Insidus Corporation
www.insidus.com
August 2004

You realize that your process cannot continue as it has for very much longer. The system that you have in place, or lack thereof, is starting to show some major cracks. Maybe your process now involves a complex series of linked spreadsheets that lack data integrity. Has your process outgrown its current database, or maybe your legacy system no longer provides the meaningful analysis that your business requires. Even if you do not currently have a formal system in place for your process, it is decision time on how to proceed. Do you purchase commercial off-the-shelf (COTS) software or do you build it? If you decide to build it, do you build it in-house or do you outsource to a third party vendor?

The build or buy decision is very important and complex. Of course your organization wants the most cost-effective, functional and feature-rich solution, but it is not always clear which decision is right for your process. Depending on your access to resources, your decision may already be made for you. For instance, if your organization does not have a development team or you do not have access to them, then you obviously cannot develop your application in-house.

In terms of quality, unless your organization or department’s core competency is building complex software applications, odds are that you will not be able to build it better than if you bought it or outsourced the project. If your in-house development team does not regularly create applications on the same scale that you require, they will grossly underestimate the time required to build the application. In this situation, your best-case scenario would be a lengthy delay in the deployment of the application and a very likely outcome would be complete failure.

When developing software, research has shown that there is an inverse correlation between the size of a project and the probability that it will actually be completed (see fig. 1.0). It’s not uncommon to hear about a large bank that lost \$7 million after pulling the plug on a \$30 million project. The likelihood of success is greater for outsourced projects when compared to in-house development, most likely because software development is the core competency of third party developers. This means that projects that are larger in scope should most likely be developed by a third party developer or you should scour the landscape for COTS application that can be customized to your requirements.

Project Size	People	Time (months)	Success
< \$750k	6	6	55%
\$750k-\$1.5M	12	9	33%
\$1.5M-\$3M	25	12	25%
\$3M-\$6M	40	18	15%
\$6M-\$10M	250+	24+	8%
\$10M+	500+	36+	0%

Figure 1.0 *

Buy

The main rationale for using COTS software is lower initial costs and a reduced time to deployment. This may or may not be true, but we will touch on this point later in this section. Ongoing per user licensing and maintenance costs until retirement may make COTS a very expensive option over the long term.

In a crowded marketplace, you must do your due diligence to identify and evaluate COTS solutions that are available. Realize that this can be a very time-consuming process, particularly if you do not

know the landscape well. Most commercial software companies provide free evaluations of their software, so make sure that you form a good cross section of future end users to test the functionality of the evaluation versions. Do not make the mistake of excluding end users. There is no worse feeling than discovering post deployment of a COTS software application that end users do not find it intuitive and refuse to use it.

The general rule of thumb is that the more generic that your business challenge is, the more likely you are to find a COTS product that fits the bill. Likewise, the more unique that your business challenge is, the less likely it is that you will find a COTS product that fits the bill. For example, if you need to share and coordinate client information amongst employees located in different departments or countries, then there are a plethora of customer relationship management (CRM) solutions available. Similarly, if you require an inventory management system or human resources system, then COTS might be a very attractive solution for your challenge. Most COTS software is very rich with all kinds of features, bells and whistles. They have to be, because COTS software applications try to meet the requirements of a very wide audience. This means that in most likelihood, your organization will only wind up using a fraction of the features of a COTS application, but you will most likely pay for them all. Since COTS software is intended for a wide audience, more than likely it will not be absolutely perfect for your organization. You may find though that the COTS application that you are considering, meets enough of your requirements that it still is an attractive option. The one word of caution here is that you should be careful that your organization is not negatively affected by having to drastically change its processes to fit the software. It should work the other way around.

For the right organization or for the right business challenge, COTS software could be your quickest and most cost-effective solution, but beware of a couple of very common COTS traps. Make sure you know all costs before you proceed. If the COTS application that you are evaluating for purchase requires extensive customization to fit your organization's needs, then make sure you factor in the price associated with that work. Many corporate customers purchase a COTS application, only to find that they have to pay a large consulting firm three times as much to come in and customize the application. Suddenly, what looked like an attractive option because of lower initial costs has turned into a very expensive low, or negative return on investment (ROI) option.

A COTS myth, which is more prevalent at the executive level than at the developer level, believes that you can create a COTS component-based system without developing. Combining COTS-based components from different software vendors to create a system does not mean that you can avoid developing. It may be a different type of development than if the system was built in-house, but you should still expect to do development. The big surprise is that you may have to do the same amount of development or even more than if you decided to build the system in-house. That could become a very expensive proposition. With a COTS component-based system, your system is always subject to the volatility of the COTS components (volatility relates to the frequency with which vendors release new versions of their products). Expect exponentially increasing volatility with time and the number of components used. If you are leaning toward a COTS component-based system, make sure you consult with your development department for their input. The comments that you receive may make you think twice about this option, particularly if your development department has not had a lot of exposure in trying to make COTS components work with each other. If your organization does not have a development department, or if you cannot gain access to the developers, then you might want to abandon any thoughts of a COTS components-based system. Outsourcing to a third-party developer might become your only viable option.

A huge risk of deciding to proceed with a COTS software application is that your organization is at the mercy of the software vendor as it relates to future product features. Changes in your marketplace or industry could have large implications on software requirements. If a paradigm shift starts occurring in your line of business, how much influence will you have on the software vendor to make extensive software modifications, and at what cost? If you are a behemoth like General Electric or Bank of America and you have a multi-million dollar deal with your software vendor, odds are pretty good that they will listen and move accordingly. If you represent a very small portion of the software vendor's revenue, odds are they will tell you that you must wait until enough of their customers have requested the feature(s) in order for development to make economic sense for them. There is not much that can be done at this point. Software vendors have to profitably run their businesses as well. In fact, you should want your software vendor to be profitable. A failed software vendor is not available to support their product. It is a frustrating feeling being locked into a product financially without any development influence so choose your COTS software vendor carefully. It may make a whole lot of sense in choosing a vendor who is responsive and willing to work as a partner to your business. Ask the right questions at the beginning. It is all part of doing your due diligence.

Build In-House

The decision to build a software application in-house becomes almost a knee-jerk reaction when you have access to your organization's development team. The arguments for doing so often times are as follows:

- We know our company's processes better than anyone else
- We can develop precisely what we need
- We have direct control over future development and can react quickly with modifications as the business changes
- When we build it, we have a complete understanding of how it works
- Since it is proprietary, we never have to worry that our competition will get it as well
- The company has already paid the sunk (fixed) cost of the development team salaries
- There are no COTS software applications that even come close to providing the specialized functionality that we require

There may be some truth to the statements above, but be careful that you do not buy into your own propaganda. It could wind up being a very costly decision for your organization if your analysis is flawed. There is a tendency to skip the due diligence phase and many times the analysis phase when deciding to develop software in-house. Accepting the above statements as fact within your organization could be quite risky. Each organization is quite unique from the next and your development department is quite unique as well. Not all development departments or developers are created equal, so you need to know the capabilities of your company's development team.

Many times when a potential project is cooked up at the management level, the only time invested in investigating whether or not your development department can create the solution is just posing the question to the developers themselves. There are two major flaws to this approach. If the choices for a programmer are to continue generating boring report requests for the accounting department at the end of each month or to create the next "killer app", which do you think the programmer would rather do? The second flaw is that there is always a massive communication gap between management and the developers. Neither group understands the other because they do not speak the same language. Management types are great at looking at the world from 10,000 feet. Unfortunately, the devil is in the details, and programmers must work with details down to the most granular level. What may appear as a small additional request from management may require two straight months of programming to incorporate into the system. Sure the programmers

promised that they could deliver the goods on day one when you presented the original project deliverables, but the continuous “scope creep” is blowing the project way out of proportion. “Scope creep” also has a tendency of rearing its ugly head during the beta testing phase and immediately post implementation.

The disconnect works in both directions though. Most programmers do not understand the business requirements. What might be painfully obvious and straight forward to a developer might not be all that intuitive to an end user and without well-defined deliverables and software design at the beginning of the project, you might wind up with something much different than you first envisioned. Even with well-scoped deliverables and design, the business unit requesting the software needs to be involved throughout the lifecycle of the development. If you think that there will be no involvement or time commitment from the requesting business unit, you are in for a very rude awakening. Be prepared for time overruns if your development team sends an iteration of development work to the business unit to review on the Monday and the business unit does not even look at it until Friday because they are swamped. You have just lost a week on this iteration and there will be numerous iterations throughout the life of the project.

When initiating a new project, you should agree upon project owners from the business unit who will eventually use the software and make sure they understand that they will have to kick in a little extra time and effort over the coming months. Many front office types seem to think that all programmers work in a dungeon, love working until 2:00 am and enjoy living on cold pizza and sodas, so it’s not a big deal to hurl one more request at them. The reality is that if you want to successfully create the next “killer app” for your organization, it has to be a team effort from all stakeholders. This may mean that even the business unit project owners may need to stay late at work on occasion.

Hopefully your organization has a process where the business units cannot simply approach the development team and monopolize their time at will. Requests, no matter how small, should all go through the development team lead or department head. One would expect this person to make informed and wise decisions regarding his development team. For larger projects, it is unthinkable to scope out a project without the involvement of the development team head right from the start. In my experience as a VP of Business Operations for a large global Investment Bank, I found that your best chances for successfully bridging the communication gap between the business units and the developers is to try to identify individuals within the business units who might have some past programming experience, work extensively with spreadsheets or at the very least are very tech-savvy. If these types of individuals become your project owners from the business side, you’ll have some great liaisons for communicating with your developers.

Most times, in-house projects proceed without any formal financial analysis. For many organizations, their view is that development engineers are already paid for, so their salaries are considered sunk costs. This is a very incorrect assumption because it assumes that you have a team of programmers being paid to sit around and twiddle their thumbs until your project comes along. In reality, if there was a lack of work for the developers, then the size of that department would shrink. Otherwise, there are opportunity costs associated with working on your project. The developers would not be able to dedicate time to other projects and tasks and quite possibly your organization would have to pass on a more attractive opportunity if it presented itself mid-way through the development. Ongoing maintenance and support costs are invisible as well. As a result, many organizations do not account for the time needed to repair bugs, add software enhancements and maintain compatibility with system upgrades. This could greatly impact your up front financial analysis to determine the viability of an in-house project as maintenance costs are about 55% ** of the total cost of the product life cycle.

It is sometimes difficult, if not downright impossible to calculate an accurate ROI for in-house development projects for the following reasons:

- Impossible to measure opportunity costs before the opportunities present themselves
- Difficult to allocate developer salaries to the project, particularly if the developers work on multiple project or have other duties
- Ongoing support costs are difficult to predict
- Ongoing maintenance costs are often overlooked and difficult to predict
- “Scope creep” and the associated “cost creep” are often ignored and difficult to predict
- Tendency to underestimate the time and resources required to design, develop and test
- Difficult to estimate a “buffer amount” to cover unanticipated development hiccups
- Future enhancement requests

Locate and review our white paper on how to calculate ROI for software projects. It outlines how you can quantify some of the costs above and calculate your ROI using reasonable assumptions.

Below are a number of real pitfalls that many organizations fall into when developing software in-house:

- Unrealistic deadlines from management
- Vague definition of project deliverables
- Inadequate time allotted for software design
- Little or no beta testing
- Internal politics
- Poor estimating techniques (time and cost)
- Lack of a quality assurance process
- Incorrect (or no) calculation of actual development costs
- Lack of proper project management
- Insufficient resources for ongoing maintenance and support
- Lack of up-to-date design capabilities
- Documentation is overlooked or avoided
- “Scope creep” and associated “cost creep”
- Developers required to perform current job and develop new applications at the same time

An organization that regularly produces successful in-house software applications will have addressed most if not all of the issues above. If you are thinking about developing in-house but your development department has not yet established a credible track record, you might want to start small and use the opportunity to build a process to developing in-house software applications that address the concerns in this section. You might also consider partnering with a third-party developer. At the very least, if your organization does not develop robust software applications on a regular basis, then consider contracting a third-party developer or consultant to project manage your first few attempts. Do not ignore the success rate statistics in figure 1.0. If developing software is not a core competence for your organization and there are no cost-effective COTS solutions on the market, then consider outsourcing to a third-party vendor to develop your unique software application.

Build by Outsourcing

For some organizations, outsourcing software development to a third party developer might be their only viable choice. If you do not have access to a development team within your organization and your business challenge is highly unique to the point where no COTS software could help, then your

choice is made for you. If you do have access to a development team within your organization, then you should perform an in-house versus outsource analysis to determine which build option is more attractive for your particular business challenge.

Obviously you will need to bring the outsourced vendor up the curve on your business practices and procedures so that they can create a robust software solution, but you would probably have to do the same with your internal developers. The advantage of working with an outsource vendor is that your management does not have to communicate deliverables directly with members of the development team. The communication gap between your management group and the vendor's development team is bridged by the vendor's software designers or project managers who are experienced in communicating with management types and they also possess programming experience so they can liaise with both groups fluently. The design and project management layer often times is missing with in-house development projects in organizations that do not develop a lot of software in-house.

Since developing custom software is the core competence of third party vendors, they will already have an efficient system for developing software in place. If your organization develops a lot of applications in-house, odds are fairly good that your development team will also have an efficient application development system in place, including a project management layer that assures proper design, documentation and testing. If your development team only develops applications occasionally, then research their processes and capabilities to obtain a level of comfort.

You must however choose your third party vendor wisely. Do not fall into a classic outsourcing trap of agreeing to a pricing structure that you are not comfortable with. If your vendor wants to charge your firm to develop an application by the hour, then I would suggest you look for other vendors. Paying for development by the hour means that you take on the entire pricing risk of the project. This can become a bottomless money pit if you are not careful. There are plenty of good third party vendors who will share the pricing risk or better yet, eliminate the pricing risk for your organization altogether by pricing out the cost of the entire project up front. This way, the vendor carries the entire pricing risk and the only way that the pricing can change is if your organization requests additional deliverables.

Below are a few benefits of using a third party vendor for development:

- You, as a customer, have the power to hold the vendor accountable for their progress and the quality of their work
- You leverage the expertise and core competence of the vendor
- A good vendor will offer innovative ideas learned from other projects
- Maintenance and support can be outsourced on an ongoing basis so that internal resources are not tied up
- "Scope creep" cannot occur unless original deliverables are altered, and the associated additional costs are immediately known
- The application can be built to fit the challenge perfectly
- Your organization can focus on its core competence
- The costs can be clearly quantified
- Opportunity to pass pricing risk to vendor so that your organization does not have to pay for development hiccups
- Outsource vendors are pretty good at estimating development times as it is part of their core competence

- Outsource vendors require well defined deliverables so you are forced to invest the appropriate time to carefully scope out the project
- Significant quality assurance testing
- Up-to-date design capabilities
- Your project does not tie up internal resources

There can be risks associated with using third party vendors as well:

- You're at the mercy of the vendor for future upgrades
- Is your vendor flexible to changes in the middle of development?
- Can they actually deliver on their promises?
- Will they offer training support?
- Are they financially stable?
- Do they require a lot of handholding?
- Can they effectively communicate with your project team members and management?
- Can they design an intuitive application that your end users will actually use?

Using a third party software developer to design and build your application can add tremendous value to your organization in the form of a highly customized application that fits all of your requirements. It can sometimes be a pretty costly option as well, but at least the costs are visible. Outsourcing in general has become very popular as it can allow you to convert fixed costs into variable costs, but beyond flexibility and cost savings, outsourcing allows your organization to concentrate its effort on your business and it leverages best-of-breed knowledge for non-core tasks.

Summary

Complex software decisions are always difficult to make. Making the wrong choice for your situation could have huge implications on your organization and your career. Do your due diligence so that you are comfortable with your choices. Even if your chosen path turns out to be the incorrect one, you can save your career by showing your executive team that due diligence was performed and that at the time that the decision was made, your organization made the best decision possible given the information.

If your business challenge is fairly generic in nature, you do not foresee a lot of customization to make the product fit your organization, the costing analysis looks attractive and you can identify COTS software products on the market that would fit your bill, it probably makes the most sense to look at a COTS software solution.

There is no silver bullet here. Invest the time necessary to make an informed choice. The key to a sound financial analysis that supports your decision is to make sure you examine all costs for the entire system lifecycle and make reasonable assumptions where concrete figures are not available (see our ROI white paper). Arm yourself with good solid information and analysis as you may find corporate politics playing a big role in determining how you proceed. Other stakeholders may have their own agendas for supporting a different path than yours, so make sure you can back up what you say.

References

- * Khaled El Emam, K Sharp Technology Inc., Software Process and Methodology Seminar, Toronto, Ontario 2003
- ** I.C. Harris, "Using an Economic Model to Tine Reuse Strategies," 5th Annual Workshop on Software Reuse, 1992