# Software: Packaged or Build

**The Guiding Principles**

By Harris Kern's Enterprise Computing Institute

IT's mission is to produce useful business applications by all means necessary. There are many tools and flexible software architectures that will make the job easier, more bug free and your programmers more productive.

I divide this article into two parts. The first is a discussion of working with packaged application software, including selecting, implementing, and managing off-the-shelf applications. The second part is a discussion of what to do when you can't find a suitable package and need to develop your own. More IT managers have gone on the rocks by failing to bring internal development programs to completion on time and on budget than any other cause. I hope to give you some tips that will help you avoid being one of the casualties.

## Make vs. buy

For many years following the introduction of computers to business in the 1950's, corporate IT departments assumed they would write most of their business software. Larger corporations would have programmers on staff to write corporate information systems, and smaller companies that could not afford staffs of their own used consultants to create data systems. Gradually, packaged software emerged first to perform standard corporate functions including accounting and payroll, and later to do other standardized functions such as purchasing, human resources, and manufacturing. As technology has improved, the packages have become more flexible so they can be adapted to individual businesses and packaged software is assumed to be suitable for most core business functions.

With the arrival of integrated enterprise data systems such as SAP many IT managers assume their data architecture can be built entirely around off-the-shelf software, and in-house capabilities can be abandoned or outsourced. Before you drop your IT capabilities, consider some of the issues in the use of purchased software and the implications for your business. Otherwise you may discover you have let your programming team go, and it will be too late to undo your error if the package doesn't meet your needs and you do need to "roll your own."

## When packages make sense

Packaged software works best for commodity functions that are slow to change and are not strategic to your business. The best example is basic accounting. No one writes accounting software any more virtually all packages have the correct core functionality built in. No business should have difficulty finding an off-the-shelf accounting solution that will meet its needs, except for organizations that are institutionally unique in their accounting practices.

There are many other business functions that lend themselves to packaged solutions. For example, payroll packages require continuous updating of tax information, and the package suppliers can maintain the tax experts on staff much more cheaply than all but the largest companies.

However, if you believe that a function is strategic, or that your business model is unique, beware the packaged solution. Many companies have suffered trying to force fit inappropriate packages into their operations.

When is a business function strategic? When it is a part of a process used to gain competitive advantage. If your company does some part of its business uniquely to be able to produce or deliver its products or services better or cheaper than its competitors, be careful of packaged solutions. For example, Federal Express developed the Powership system to allow customers to quickly prepare their goods for shipment, and to quickly track packages through out Federal Expresses system. FedEx spent millions developing a unique shipping system, both because there was no comparable system available for purchase, and Powership gave Federal Express a competitive advantage over UPS. If you are using a packaged solution, the business practices embedded in that package will be available to any competitor.

Another reason to be cautious of purchased solutions is if your industry or business model are either uncommon, or rapidly changing. Commercial software packages attempt to capture best business practices in their solutions, but they must also keep one eye on their legacy customers and on the practices common to the vertical industries they serve. As a result, if you want to move aggressively ahead in implementing a new technology or business practice, you need to either carefully select a package that will enable that practice, or be prepared to write your own solution.

Another reason to consider a software package is that it could be an important part of a strategy to use the package to energize business practice improvement in the organization. The principles are simple – find the software package that comes closest to the business practices you want to implement and then use the package to force the operating business unit to change its processes to fit the package's requirements.

## Evaluating Packages

Selecting enterprise-packaged software is a difficult and time-consuming task. It is not uncommon for companies to assemble task forces that will spend months on a package selection, or to hire a consulting firm and spend $100,000 to pick a package. This selection briefly discusses some of the processes and selection criteria needed for a successful package selection.

### Selection criteria

Every company needs to develop its own requirements for the selection of an enterprise software package. But many of the issues in selecting a package are common to everyone. This list will seem sophomoric, but I have seen many horror stories of companies that were naïve in their purchases and paid a heavy price in implementation.

These issues include:

**Cost**
Purchase cost must be carefully evaluated since software vendors have different pricing models. Some vendors' price by connected user, others price by named user (even if not connected). Some price by platform with installs on MVS mainframes priced higher than systems on NT, even if the number of users is the same. Also, be sure to ask for the total cost of the software from pilot to full deployment or you may be in for a shock.

Your cost must include annual maintenance and support fees, anticipated during the life of the system. These recurring fees must also include anticipated one-time upgrade charges if applicable. The price of development and maintenance tools must be considered, if not packaged, with the software.

If you are considering buying a total enterprise solution, carefully consider the cost of other modules you may want n the future. If you are buying a general ledger system will you also want a cost accounting or purchasing package from the same vendor? And don't forget the cost of consulting and support.

**Speed and ease of installation**
It is hard to quantify the costs of speed and flexibility, but they are very real and worthy of due diligence with any prospective software vendor. Depending on the sophistication of your IT organization, there can be significant differences in the ease and speed of installing different packages. Some are tightly integrated in their functionality, while others can be more easily installed in modules. Some packages have better GUI-based tools to ease setup and maintenance. Some require learning a proprietary language or database. Others may have particularly long training periods. Talking with recent customers and reading the literature can be helpful in bringing these issues to the surface.

**Functionality**
The old KISS rule is still the best – keep the package as simple as needed to do the job. Some packages have limited flexibility while others have every feature you may ever want. But remember that all those features that make your customers say "gee whiz" must all be maintained, users must be trained, and hardware must be sized to accommodate the features. A colleague recently installed a new financial package in a division of a large natural resource company. He picked a fairly basic package, with simple functionality. At the same time, another division selected a more complex and feature-rich package from a competing vendor. The simpler package was installed in two-thirds the time for half the price.

**Technical and architectural issues**
Packaged software vendors have different approaches to their architecture, and you need to be sure their architecture is compatible with yours. Some vendors provide API's that will allow you to bolt on specialized or legacy systems. Others provide proprietary languages to customize the screens or functionality. Some vendors have built their systems with object layers to enable higher levels of flexibility between modules or between their systems and other vendors. Here is a quick checklist of issues to discuss with every prospective package vendor:

- Support for open standards, popular operating systems, and hardware

- Development tools for modifying the data structures and input screens
- API's to connect other packages or home-grown systems
- Support for popular syste3m management tools i.e., Tivoli
- Utilities for backup, performance management, and system administration
- Integration among modules for enterprise packages, and the coherence of the upgrade strategy among modules

**Selection process**
Force-fitting a package into a reluctant customer organization is often a pyrrhic victory. You may win the battle, but lose the mind share war with your customer.

The essential challenge is to involve the functional customer groups to gain their buy-in, while not confusing them with technical issues. More importantly, you need to keep your credibility and customer focus so that you can counteract the impact of "gee whiz" features and clever vendor selling. This process can be daunting and time consuming. However, if you don't have the time, but have some money in your budget, every major consulting firm has a package-selection practice and can help you and your customers buy wisely. If you want to do it yourself, the following is a simple roadmap of the steps in package selection:

**Initial requirements selling**
Setting functional and performance goals for any new data system is the most difficult task. It is axiomatic that customers always interpret their needs in terms of what they know, and I have seen customers drive decisions toward packages that look like their existing systems but with the rough edges knocked off. It quickly becomes IT's job to intervene to be sure that the enterprises needs are being met and that the new system has the functionality and flexibility to be integrated into the corporate architecture.

In setting requirements, be sure that all participants understand the reasons for shopping for a new package. Are their functional deficiencies that will no longer support the business model? Is the existing system too old and cost too much to maintain? If you are replacing a package, is it because the existing vendor is failing to keep up with evolving technology? Each of these reasons can lead to a different selection criterion and different level of customer satisfaction. Functional issues often must be reengineered into the customer organization, while technical issues can often be left to specialists.

**Selection of candidates**
When the requirements have been decided, a *Consumer Reports*-style chart should be produced for each candidate vendor summarizing how well each vendor's capabilities maps into your requirements. At this step, disqualify any vendor that has clear deficiencies. Don't be swayed by vendors selling futures or spreading FUD. Your goals are to reduce the number of packages to two or three candidates worthy of serious consideration. The more you let into competition the harder the evaluation and selection process will be. Complexity grows exponentially with the number of vendors. Now is the time to be tough. Weed out weak or inappropriate vendors early.

### Scripted functional demonstrations
Scripted demos have become a popular way to allow your functional customers to become familiar with the capabilities of the package, while learning more about their own needs as well. Scripting, the act of creating a roadmap for the demo is essential. Scripting captures the requirements of the customers and inhibits the vendors' natural inclination to play to their strengths and avoid discussion of weak or missing features. Every customer has a wish list of needs, But as they run through the scripted demos customers begin to learn the trade-offs vendors have made, and they will also learn which features are "must haves" and which features they can live without.

This is also the time for you to begin learning about the vendors. You are going to be making a long-term commitment to the vendor's people and organization as well as their product, so now is the time to find out how knowledgeable is their support staff, how responsive are they to issues that arise, how technically sound is their product. This discussion even goes to chemistry issues. Do you approach problems the way your organization does. Are their people empowered? Do you have access to management when you need it? If they don't impress you now, they won't likely impress you later.

### Technical evaluation
While the functional customers are going through the demonstrations discussed above, a technical team needs to carefully evaluate the package architecture using the criteria discussed above. The goal of this process is straightforward – determine whether you can live with the package in light of your architecture and technical vision. If not, don't touch it.

### Business discussion
In deciding on a package to become a part of your enterprise decision, remember that you are not buying a point solution, you are buying a long-term relationship. Be sure you clearly understand purchase costs, maintenance fees in future years, training costs, upgrade costs, technical support costs, and what insurance you have in the contract to enable you to control those costs. Get all the costs on the table, including upgrades to your infrastructure, before you decide. Going back to management with a bunch of gotchas later is not career enhancing.

### Decision
Once the decision is made, check to see how loud the screaming is from those in your organization. If people are comfortable with the decision and anxious to move ahead, the decision is right. If you are going to have to drag some departments kicking and screaming, then either ensure their support, the willingness of the executive sponsors to push with you, or don't move forward. If you are out too far in front of your organization on the issue, you might think about investing more in helping your customers prepare for the future. IT management is hard enough without dragging your customers too.

## Living with Packages
Packages software is advertised as the low-cost, simple, solution, but there are collateral issues that need to be considered. First, the package vendor now controls the introduction of new functionality. If internal customers request changes to a package function, they may have to wait or if the requested feature is not in the vendor's product plans, do without.

Second, do not underestimate the requirements for upgrades and maintenance. A major revision to a package can take days or weeks. Some vendors do not (or may not) use common data formats between releases. The result is that data may need reformatting and file structures completely rebuilt. Most vendors will supply conversion tools, but the job can be time consuming and in the case of large enterprise systems like SAP, considerable advance planning is required for a successful upgrade or revision.

The final issue of living with packages is that you must assume the business risk of the package vendor. If the vendor goes into decline or bankruptcy, or is merged or sold, how will you maintain and improve the package and at what cost? The risk can be mitigated but never eliminated.