Build v. Buy

A Decision Paradigm For Information Technology Applications

By Kenneth S. Ledeen,
Chairman and CEO, Nevo Technologies, Inc.

NEVO

www.nevo.com

Few Information Technology topics have received the sustained interest visited upon the question of whether organizations should acquire commercial off-the-shelf (COTS) software packages or build from scratch. No general answer exists to this complex question, but a solid understanding of the differences between the two, and a structured approach, can remove both the uncertainty and the risk from this critical decision.

Not only have opinions varied, but also the balance of thought has tended to move from side to side, sometimes favoring one approach, then the other as technology advances. The emergence of highly configurable, sophisticated applications in such areas as ERP, Human Resources, finance and administration, CAD, CRM, Business Intelligence, and Data Warehousing fueled the popularity of packaged solutions.

Recently, the Gartner Group[i] reported on a growing trend towards "build" citing several drivers for this trend: the impact of new technologies and the availability of skilled technologists, the recognition that many packaged solutions are "too cumbersome, bloated and expensive," the need to "adapt to unique business models and the idiosyncrasies of their own organizations."

For many applications the line between build and buy has become blurred. Custom applications are often created by integrating standard components, and packaged applications often require extensive configuration and integration as well. Building custom apps today is more a process of assembly than construction. The distinction is often a matter of degree.

This paper presents a structured decision making paradigm that takes into account both the nature of the organization, the processes under consideration, the technology environment, and the state of packaged solutions and their vendors. It yields sufficient information to allow a selection team to make an informed decision.

We will present a set of decision criteria, discuss the inherent differences between packaged solutions and custom ones, and finally, outline a structured decision process.

I. DECISION CRITERIA

Many build v. buy decision paradigms share consideration of these important factors:

The Criteria

- Core vs. Context
- Coverage
- Direction
- TCO
- Scale
- Timing
- Standards

Core vs. Context

Borrowing the terminology from Geoffrey Moore[ii], the first criterion is the strategic significance of the application. Applications that do not impact the unique nature of the business rarely warrant the attention that custom solutions demand. Few organizations would consider custom software solutions for general ledger, HR and payroll, tax preparation, or supply chain management. But for H&R Block, tax preparation is core to the company's strategic advantage, just as for WalMart, supply chain management drives their success.

Core activities are those that contribute directly to the organizations differentiation and value creation. Context is everything else.

It is in Core areas that organizations gain strategic advantage, and where information systems must conform to business processes, not the reverse.

The chart below, borrowed from Geoffrey Moore, speaks to where organizations gain the greatest leverage and strategic advantage through investment of intellectual and financial resources. Mission critical, core activities are the highest priority and deserve the most attention. When assessing build vs., buy decision, know whether the particular activity is core or context will suggest whether it is appropriate to consider modifying business practices to meet the strictures of a commercial package, or whether the best needs to maintain direct control.

	CORE *ENGAGE*	CONTEXT *DISENGAGE*
MISSION CRITICAL *CONTROL*	MAKE	OUTSOURCE
SUPPORTING *ENTRUST*	PARTNER	CONTRACT

Coverage

Coverage assesses how close the match is between what the business requires and what the packaged solution provides.

The traditional rule of thumb is that packaged solutions must meet a minimum of 80% of the required functionality. Unfortunately, this is an oversimplification.

It is not sufficient to determine only if a package covers all the requirements. It is equally important to determine if the package provides capabilities that are outside the requirements. Every feature, every function, every capability represents additional cost and complexity, and those inevitably translate into future costs and complications. Just as it is unwise to consider a package unless it meets 80% of the known needs, similarly, it is unwise to consider one in which the known requirements represent less than 80% of the package's capabilities.

A common trap is evaluation COTS-based solutions solely on the features they offer. This feature-checklist approach offers a simple mechanism for comparisons among packaged alternatives, but misses the key dimension of business process fit. Feature lists rarely capture the dynamic characteristic of method and process.

Requirements must be compared from the perspective of business process, not exclusively from features and functions. One of the most common sources of problems down the road is a reliance on feature-laden technical specifications.

All too often packages will meet the functional requirements, but fail to provide a solution consistent with the organizations business processes. Or, vice-versa. A potential solution is ruled out because it doesn't match the exhaustive feature list, but would have been and excellent for the business process.

Direction

In assessing packaged solution, the concern imust be not only how well current requirements are covered, but also how flexible, maintainable, and extensible the application will be throughout the intended life of the software. This is particularly important for applications that are intended to have a long useful life since many of their, requirements have yet to be imagined, let alone defined. When we consider direction we must take into consideration how much control the organization will have over the crafting and addition of desirable, but as yet unknown, new features.

The consideration of direction applies equally to application functions as well as to platform technologies. Both must be considered carefully. Architecture, technology, integration methods, all influence the long term direction and evolution of the package. They are key indicators where the package will be, how long it is likely to be enhanced, and consequently, how likely it is that the solution will continue to be a good fit in the future.

Total Cost of Ownership (TCO)

TCO includes not only the cost of acquisition, configuration, and customization, but also the ongoing support, maintenance, and evolution of the application. It is quite common for lifetime costs to dwarf acquisition costs. Perhaps the most important determination in the calculation of TCO is an estimate of the anticipated economic life of the application. TCO must take into account the fixed costs associated with mastering the underlying structure and technology of a packaged solution as well.

Custom solutions required continued availability of development resources, either in-house, or through partners, to respond to changing requirements. Similarly, packaged solutions require resources to test, validate, integrate, and support new releases from the vendor. The degree of effort is highly dependent on the nature of the application itself.

Typically, packaged solutions have much higher volatility – that is they tend to change more often and in more dramatic ways – and a much shorter economic life.

Both of these factors are discussed in more detail below.

Buying a solution that has hosts of capabilities that you don't need is a bad idea. One way or another, you end up paying for the complexity in terms of training, integration, configuration, maintenance, support, or any of the myriad factors that influence long-term costs. Beyond the pure economic impact, the vendor will be revising, enhancing, and expanding all of those additional features as well. If the package does so much more than you want and need, then it is likely that it is meant to serve a different audience, and your requirements stand a high probability of diverging over time.

Scale

The larger the scale of the application, the more important it is that it represent core business functions. Conversely, many large, integrated ERP solutions have such a large fixed support cost that it is not reasonable or appropriate to consider implementing a small subset of the complete suite.

Scale becomes an important factor in measuring and ultimately mitigating risk for the project. The decision process must yield both a comparison of costs and risks in order for informed decisions to be made.

Timing

Conventional wisdom held that implementing a packaged solution was faster than custom development. As one might expect, this is an oversimplification. The process of installing, configuring, customizing, and completing data conversion for packaged solutions routinely involves tasks that are as complex and extensive as custom development, with far less flexibility in phasing and timing.

COTS packages may offer greater predictability with respect to implementation time, but that is largely a reflection of the restrictions they impose on capabilities and flexibility. The greatest risk to timing in custom development is the difficulty organizations have in control the requirements process and allowing feature creep to occur.

The greater the degree to which the organization can accept a pre-defined business process, the simpler the implementation will be. If a package can be used "as is" without any adaptation to the organization's business processes and practices, then it will have a substantial advantage over a custom implementation. As soon as "configuration" is required, or business process modification is considered, packaged solutions cease to have any meaningful time advantage. They merely trade one form of activity for another.

Standards

Standards may be the most important criterion of all.

COTS vendors market the notion that the cost of software development can be spread across that a large user community, thereby reducing the cost to each individual customer.

For this promise to be true, there needs to be some external force that ensures consistency of at least a large portion of the requirements. Common examples of such forces are government standards (tax laws, accounting regulations), nature (software to do simulation, for example), broad industry standards (HTML standards allow commercial browsers to exist, ISO standards for mechanical drafting), or powerful industry groups (FIX for financial transfers, EDI, AIA standards for architecture, and others).

Standards may also arise directly from the success of a particular package in areas where organizations are highly likely to see the function as "context" not "core. Office applications (word processing, spreadsheet, email clients) are good examples. Tools and components also fall into that category. Few organizations would think to develop their own database technology, messaging middleware, or operating system.

In the absence of a strong external force defining consistent content, it is nearly certain that COTS solutions will diverge over time (see "Direction" above); that the feature-function evolution will cease to be a good fit (see "Coverage"); that the TCO will grow as organizations cope with a mismatch between the package and the business requirements. Worst of all, the business users may discover that they cannot ensure that evolving software will match their business process requirements – particularly for functions that are "core."

II. DIFFERENCES

While the decision help us evaluate fit, we also need to consider he inherent differences between packaged and custom solutions.

The Differences

1. Economic Life
2. Volatility
3. Business Process
4. Timing
5. Control

Economic Life

Custom solutions typically have a substantially longer Economic Life than Packaged COTS Solutions[1]. Two factors contribute to a difference that can be ask great as 3X: the complexity that comes from diverse requirements, and the pressure to use the latest technologies.

First - complexity. Software has the unfortunate characteristic that the more complex it becomes, the harder it is to extend, modify, or support. COTS solutions become the victim of their own success: more customers means greater diversity of requirements. The complexity needed to support that diversity shortens the time over which they can meet evolving needs because it becomes to difficult to enhance. It is not uncommon for packaged applications to have a data model with many times the number of tables of custom solutions.

Second - technology. New customers of packaged solutions demand that those solutions be implemented using the latest and greatest technologies. This is not at all unreasonable, given that new customers are looking for new, advanced solutions. However, technology platforms are utilitarian for much longer than they are popular. No organization would choose to acquire a COBOL-based application today, yet large systems in many industries continue to function perfectly well with them. The result is pressure on COTS vendors to migrate to newer technologies faster than the underlying system requires. If they don't, they will become uncompetitive. Thus, it is the requirements of the "next" customer that drives the actions of package vendors, rather than the needs of the user base.

The combination of increased complexity and technology migration results in an economic life for packaged solutions that is one half to on third of the expected life of custom solutions. It is not at all uncommon to encounter custom solutions that remain highly productive ten, fifteen, or twenty years after their implementation.

Volatility

By "volatility" we mean the frequency and complexity of new releases. Greater volatility means increased support and maintenance costs, since a common characteristic of packaged solutions is that customers must test, integrate, and install each release, whether it contains desired enhancements or not.

Each new release presents substantial risk to the stability and availability of the system to users. The burden of validating content, testing

Custom solutions, by contrast, are only modified in response to user requests or changes in the client environment.

Business Process

While it is common for packaged solution vendors to claim complete flexibility in configuring their solution to meet existing business process this is rarely the case. Most often client organizations are urged to modify their business practices to conform to the range of choices that the package offers. This constraint is essential lest the package grow to be unmanageably complex. In practice, this is often not a problem since one of the motivating factors behind the choice of a packaged

[1] *COTS-based System and Make vs. Buy Decisions: the Emerging Picture,"* Abts, Chris, Center for Software Engineering, University of Southern California, Position Paper for the International Workshop on Reuse Economics, Austin, Texas, 4.16.2002

solution is the desire to re-engineer business processes.

3. DECISION PROCESS

The Build v. Buy decision paradigm we advocate seeks to understand the business needs, and then eliminate candidate solutions that fail to meet those needs. The remaining choices, all considered "acceptable," are then compared from the balanced perspectives of cost and risk. To accomplish this, we use a six-step process:

The Process

1. Assess Organizational Bias
2. Determine Core vs. Context
3. Document scenario-based requirements
4. Review available packages for fit.
5. Develop TCO estimates for all best-fit alternatives
6. Prepare Risk/Mitigation Matrix

Assess Organizational Bias

We start here because all organizations have an innate bias towards COTS or custom solutions. That predisposition should be acknowledged, and the reasons for it understood so that they can be addressed directly. The position may be well founded, or based on past experiences that are no long directly application.

If organizational bias is not understood and made explicit, the assessment team can waste substantial time and effort and not provide the information needed to reach a reasoned conclusion

Determine Core vs. Context

It is quite common for organizations to lack clarity in this area, or to confuse traditional business processes with strategically significant ones. Since the choice of a COTS solution will almost certainly result in the transition from current processes to the ones the package supports - at least at a detail level – it is extremely important to gain insight and concurrence on the strategic significance of the business processes the application will support.

There is ample evidence to lead us to a COTS solution for context activities, and a custom solution for core, differentiating activities.

Document Requirements

We strongly advocate the use of scenario-based requirements, supplemented by functional requirements where appropriate. This approach (called Use Cases in UML) ensures that the chosen solution will meet the business objectives. Requirements that are limited to a Functional Specification miss all of the dynamic aspects of system usage.

At this stage of the process, it is useful to approach the process as if you were going to build the whole thing, documenting what you want and need in a way that would let a development team understand what they are to build. Avoid simply prioritizing features and functions, and stick to scenario-based requirements supplemented by high-level, non-functional needs (such as security, availability, access).

Review Packages for Fit

This is where the bulk of the work gets done. Concentrate on both coverage and direction since you need to be sure that the package will meet not just your current needs, but your future needs as well.

When assessing coverage, consider that the 80% rule should apply in both directions – both what is in, and what is out. A package is a bad fit if either

- It doesn't meet 80% of your needs, or
- Your needs represent less than 80% of what the package does.

Finally, fit should be measured in the non-functional domains as well. The underlying technology, support practices, frequency of release and upgrades, integration tools and capabilities, are among the characteristics worthy of assessment.

TCO Estimates

The economic phase of the determination requires a solid understanding of the total cost of ownership, not simply the cost to acquire, configure, and implement.

TCO calculation needs to take into account the startup costs, as well as the economic life and volatility of the package. How far through its life cycle is it? How often are major and minor releases issued? Is it mandatory to test, integrate, install, and support each of these releases?

It is common for packaged solution to have very large fixed support costs, many of which are not discussed during the sales cycle, and not at all visible until far into the implementation phase.

Packages that involve substantial configuration, for example, may require elaborate and costly efforts to maintain control over the configuration as it evolves overtime.

The uncertainty that is often associated with the acquisition cost of a custom solution can be overshadowed by the uncertainty and magnitude of the lifetime costs of a packaged solution.

Risk and Mitigation

All projects involve risk, some less and some more. Each of the “acceptable” alternatives should be the object of a detailed risk and mitigation review. You may discover, for example, that the “build” alternative can be broken into smaller phases that will both reduce the risk of a big bang implementation, and give the organization time to adapt to the new, more efficient business processes.

Organizations face the build v. buy dilemma every time they turn to information technology to gain efficiency, improve productivity, or improve their strategic advantage.

Understanding the differences between the two approaches, and embracing a structured, disciplined decision-making process can yield very large benefits.

.

References

[i] ***A Build Mentality Is Re-emerging in Business Applications***, Gartner Groups, Strategic Planning, SPA-20-7537, S. Nelson, **Research Note** 21 August 2003

[ii] **Living on the Fault Line: Managing for Shareholder Value in Any Economy**, Moore, Geoffrey, HarperCollins, 2002