Build or Buy?

Edited By:

William Indest

Introduction

This paper explores several key financial issues relating to the build vs. buy decision in an attempt to explain why almost 75 percent of companies would prefer to license packaged applications or suites rather than develop their own.

A key theme throughout is that organizations making the build vs. buy decision can maximize citizen/shareholder value by focusing on the DQ project's expected return on investment (ROI).

ROI is the ultimate financial measure of investment success because it comprehensively factors in costs, benefits, risk, and the time value of money. Organizations that perform ROI analysis while making the build vs. buy decision are able to project the ROI and total cost of ownership (TCO) of a DQ solution under both the buy and build scenarios and compare the buy and build approaches despite differences in capital expenditures, project plans, and investment time horizons. Experience has shown that TCO, functional capabilities, and project risk are the three most critical factors driving the return on DQ initiatives.

TCO

The first factor driving the return on DQ initiatives is TCO. To fully explore the impact between TCO and ROI and the key cost categories for application development is not a goal of this missive. Let's just say TCO and ROI are inversely related – High TCO drives low ROI and conversely low TCO drives high ROI or… your customer wants low TCO.

Companies contemplating an internally developed DQ application project can use TCO analysis to better understand application development costs and the most critical cost drivers. The five most important cost categories for internal application development projects include design and development, integration, training and support, maintenance, and opportunity costs. You should mention this to your advocate.

Some organizations believe that they can internally develop DQ applications at a fraction of the price of packaged solutions. Strictly speaking, they may be correct. In most cases, however, the development costs saved are merely deferred and incurred at even higher levels during the maintenance phase. It is a classic "pay now or pay (much more) later" situation. The design and development of a high-quality internally developed DQ application involves a cadre of dedicated resources with significant DQ domain expertise, including developers, project managers, business users, quality managers, and supporting third parties. These resources are costly, but necessary to ensure the proper design and development of the application's architecture and data model.

An application's architecture is probably the single most important factor in reducing an application's TCO. An optimal architecture is flexible, scalable, multi-tiered and performs satisfactorily with massive amounts of data. As many organizations know, developing such an architecture can be challenging, time-consuming, and perhaps most important, costly. Unfortunately, the cost of not developing a well-designed architecture is even greater. Maintenance costs of poorly architected applications are dramatically higher than those for well-architected applications, and performance, deploy-ability, and scalability typically suffer as well. The end result is an expensive, ineffective system that delivers a poor user experience.

Application functionality is one of three key factors that affect the ROI of DQ initiatives. Application functionality directly impacts ROI because most of the business benefits (revenue enhancements and operational cost reductions) are derived from specific functional capabilities. This direct impact is discussed in greater detail below. Application functionality deserves mention in this TCO discussion because it also indirectly impacts ROI as an important driver of design and development costs. As a general rule, incremental application features and functionality increase development costs because they add complexity. According to one of the most commonly used software cost models, the Constructive Cost Model (COCOMO, [really, you can google it]) application complexity is a primary driver of software development costs. Organizations must therefore budget and spend significant funds on application development if they want to develop functionality rivaling that of packaged applications, particularly EDQ. Many organizations realize too late that the functionality they can afford to build themselves both fails to adequately support business requirements and suffers from poor user adoption, waste waiting to happen.

Effective and efficient integration capabilities are critical for both internally developed and packaged DQ applications, which, by necessity, must be interoperable with numerous other applications and we are talking about a lot of interfaces spanning applications of various types. While organizations cannot avoid integration costs, they should carefully monitor and minimize them as much as possible.

Companies that internally develop DQ applications often find that integration costs are difficult to reduce because there are few, if any, methods of reducing the required time, training, and resources. The system under development must integrate with potentially hundreds of other systems, each of which requires specialized coding. Developers must be knowledgeable about each application being integrated and trained on the various interoperability tools that are required to complete the integration. That's a bummer.

In contrast, packaged applications often enable organizations to reduce integration costs through prebuilt adapters, connectors, and integration tools. These tools are often complemented by alliances between application vendors and systems integration firms, which develop and offer validated integration solutions for the most common systems. These turnkey solutions permit rapid, low-risk integrations by experienced professionals.

Among the most commonly overlooked and underestimated application costs are those for end user training. A common misperception is that a small number of users implies a low training cost, but many organizations fail to realize that certain training setup costs are relatively fixed regardless of the number of users to be trained; materials must be developed, and trainers must be trained before users can attend training sessions.

Internally developed DQ applications tend to have higher training costs than packaged applications partly because they are often designed with a greater focus on functionality than usability. Developers are typically charged with developing a system that includes certain required functionality by an agreed-upon deadline; usability requirements and testing are often regarded as issues of secondary importance. As a result, internally developed applications often suffer from complicated views, poor navigation, and other impairments to application usability, which translate into longer training times, higher training costs, and lower user acceptance.

Packaged applications provide two benefits not offered by internally developed DQ applications that help lower training costs. First, packaged applications are designed to maximize usability because they must have appeal and acceptance in the marketplace at large and across a broad variety of industries. To help ensure that they meet market needs, application vendors leverage extensive user experience studies while designing and enhancing applications. The resulting navigational and user benefits enable organizations to more easily train users in a shorter amount of time and at a lower cost.

The second benefit associated with packaged solutions is the availability of third-party training resources. Packaged applications that enjoy even moderate market success provide consulting firms, systems integrators, training firms, and application vendors themselves with a financial incentive to develop standardized training programs. Unlike organizations that develop their own training program, these third parties are able to allocate their program development costs across all their clients, enabling them to provide high-quality training programs at competitive rates.

Maintenance costs are a critical component of TCO for DQ investments. Indeed, the majority of ownership costs are incurred not during development, but in operating, maintaining, and upgrading the system over a period of time. Organizations commonly spend between 70 and 90 percent of the funds budgeted annually to internally developed applications on maintenance (for example, application and database administration, software patches, and upgrades) and the balance on the development of new functionality. Maintenance costs for internally developed solutions are higher when organizations use a fourth-generation language (4GL) or fail to invest adequate time and resources in the architecture and/or data model during a system's design and development stage.

Opportunity costs should be included in any ROI and TCO analyses. Companies that consider opportunity costs while making the build vs. buy decision may very well conclude that reinventing the wheel can become prohibitively expensive from an opportunity cost perspective. Although opportunity costs can take many forms, those relating to time to market and organizational resource allocation should be included in every build vs. buy financial analysis. Organizations generally maximize citizen/shareholder value when they accelerate their time to market and allocate organizational resources according to their best use. When organizations fail to perform in this manner, they incur opportunity costs by foregoing value they would have otherwise earned. The most common opportunity costs—those relating to time to market, human resources, and organizational funds— are discussed separately below

While they vary in their functionality and effectiveness, most internally developed and packaged DQ applications deliver significant financial benefits in terms of revenue enhancements and cost savings once they are operational. It follows, however, that each day spent designing, developing, and implementing a DQ application beyond the optimal implementation period represents an opportunity cost in terms of lost financial benefits. Time to market therefore becomes a critical issue in the build vs. buy decision, particularly when the solution has been deemed mission-critical to a broader Data Quality strategy being executed.

Internally developed DQ applications often require longer times to market than packaged solutions because they must be designed, developed, and implemented. Given that five full-time developers might take a year to develop, say, a matching/de-duplication system,a robust homegrown profiling, standardization, matching, de-duplication, SOA-enabled solution that offers functionality rivaling that of a packaged application would likely take considerably longer to build. Packaged DQ applications, in contrast, generally reduce time to market because they enable companies to skip the design and development phases and immediately begin implementation, which can often be achieved in a matter of months.

Among opportunity costs, the time to market opportunity cost is perhaps the easiest to quantify. The expected daily financial benefits from the operational DQ application should be multiplied by the difference between the number of days for an internally developed solution to become operational and the number of days for a packaged solution to become operational.

Furthermore, companies that internally develop a DQ solution sometimes attempt to reduce the time to market by reducing the functionality offered. This approach leads to an opportunity cost of a different nature, however, because the system-generated revenue enhancements and cost reductions typically decrease as functionality decreases. These foregone financial benefits must be considered against any benefits achieved by reducing the time to market.

Internal development of a DQ application requires a sustained and dedicated effort by a broad variety of executives and employees. In these instances, the foregone value represents an opportunity cost of the internally developed DQ application.

A fundamental strategic tenet states that companies maximize shareholder value or provide better service when they capitalize fully on their core competencies. Adhering to this strategic principle requires a company's executive management to focus squarely on the profit-generating or service oriented activities that the organization does best. For companies whose core competency is not DQ application development, it follows that such development is strategically and financially imprudent, given the long-term executive sponsorship, commitment, and attention that such projects demand. Companies that elect to internally develop their own DQ applications are inefficiently shifting allocation of one of their most precious resources—executive management—from a core to noncore business activity. In doing so, executives needlessly divert their attention from a rapidly evolving workplace and marketplace to re-create a fraction of the functionality offered by packaged applications. To eliminate or minimize this opportunity cost, executives should concentrate on enhancing their firms' core competencies and leverage packaged applications developed by firms with a core competency in DQ applications.

Similarly, internal application development projects generate opportunity costs when they divert IT professionals from their best use. With technology changing and evolving at unprecedented rates, most organizations find that hiring and retaining highly qualified IT professionals is extremely challenging. Once companies identify the best and the brightest individuals, they should deploy them in the most beneficial manner possible.

At first glance, the deployment of the most capable IT professionals to build a mission-critical DQ application might seem like a good use of an organization's resources. Indeed, it would be an excellent use of resources if robust, fully-functional, easy to use packaged DQ applications were not generally available in the marketplace. Given that such packaged applications are available, however, companies should seriously question the wisdom of dedicating their most precious IT resources to the long-term development of solutions that they can readily license. Organizations incur an opportunity cost to the extent that these IT professionals could be allocated to a better use—such as developing a solution that is not available in the marketplace. Further increasing this opportunity cost is the fact that organizations must ensure that the application developers—the system experts—are available to support the application over the long-term.

Finally, organizations incur opportunity costs when they do not allocate their funds to their best use. For this reason, organizations routinely perform financial analyses to select the particular projects that will provide the greatest returns and maximize shareholder value. While making the build vs. buy decision, organizations should perform ROI and TCO analyses for both internally developed and packaged applications and then select the most beneficial approach. As discussed here, packaged solutions fare better than internally developed DQ applications over the long term because they offer the greater business benefits and lower ownership costs.

These additional revenue enhancements and lower costs not only drive the higher ROI and lower TCO results for packaged applications, but they also represent opportunity costs for internally developed DQ applications that are rarely captured in financial models. The lost revenue and higher costs associated with custom DQ applications represent foregone value that will never be used to expand or enhance an organization's core competency, be allocated to new profit-generating ventures, or be invested in the development of additional competitive advantages or modernization efforts. Organizations do not need to quantify these opportunity costs to know that they are too high, but their mere existence causes one to pause.

Application Functionality

The second critical factor that drives the return on DQ initiatives is application functionality. Indeed, it is a DQ application's functionality that increases an organization's revenues, service levels, and customer retention rates while reducing operating costs. It is these business benefits that offset and exceed the TCO to generate an ROI and improve an organization's bottom line. Most organizations therefore wish to implement and utilize as much of the functionality as possible to maximize the business benefits.

Organizations that build a DQ solution, however, often find that cost is a significant barrier to the functionality they would like to develop. As previously mentioned, incremental application functionality increases application complexity, a primary driver of software development costs.8 Organizations must therefore decide whether to build required, complex functionality despite the cost or reduce application functionality to meet budget constraints. Those electing to reduce application functionality often succeed in meeting their budgets, but pay the opportunity cost of settling for less functionality—the foregone business benefits they would have achieved with greater functionality.

Compared to internally developed solutions, packaged applications typically offer much greater functionality for use across an enterprise at a competitive cost. Organizations can maximize the financial benefits accruing from their DQ application and strategy by deploying prebuilt functionality – faster and more efficient and more cost effective.

Risky Business

The third major factor that drives the return on DQ initiatives is project risk. Internal DQ projects are fraught with business and technology risks, each of which can translate into project delays, cost overruns, and unanticipated development and maintenance. These risks can directly reduce a project's ROI because if, and when, the risks materialize, they can increase the TCO, reduce the financial benefits, or both.

Moreover, the risks are highly relevant while projecting ROI during project planning because the discount rate used to discount the expected project cash flows (costs as well as financial benefits) measures project risk and increases or decreases proportionately with project risk. Like the relationship between a project's TCO and ROI, there exists an inverse relationship between the discount rate and a project's ROI: Assuming benefits stay constant, the projected ROI decreases as the discount rate increases. Some of the most common business and technology risks associated with internally developed DQ applications are discussed below.

It is sometimes said that internally developed applications fit an organization's business needs better than packaged applications because they are designed and developed specifically for the particular business in which they are deployed. In reality, however, packaged applications are often better aligned with a company's current and future needs than even the most robust internally developed applications.

One reason for this result is that many internally developed applications are designed and developed with little input from business users. Identifying and specifying the business requirements are time-consuming tasks that are properly performed by the business users themselves. All too often, however, business users are not 100 percent dedicated to the internal development project and spend a mere fraction of their time working with the development team. Facing aggressive project deadlines, the development team does its best to identify and develop the required functionality. Without the appropriate business experiences and requisite knowledge about the business processes, however, it is severely at a disadvantage in its ability to develop an application that meets current business needs, let alone future needs.

In contrast, packaged solutions are designed and specified by professional product managers that gather business requirements from a broad customer base and then work with software engineers to incorporate those requirements into the application. This approach is a benefit in and of itself, but it leads to a second benefit that is perhaps even more important. While working with prospective and existing customers to gather requirements, product managers document the practices of, and feedback from, a large number of organizations. They then identify best practices and incorporate them into the commercially available application. Organizations that internally develop DQ applications benefit from neither professional product management nor the experiences and practices of other companies. While such applications are sometimes well-aligned with business needs, they often fall short of the mark and/or impair the very business they are intended to improve.

As previously discussed, performance and scalability are highly dependent upon an application's architecture, data model, and overall design. An inflexible architecture, for example, might meet current business needs, but struggle in its ability to bear changes as the company, its customer base, and the industry evolve. This point is critical because determining future needs at design time is among the most vexing challenges for organizations. Many companies that are currently saddled with inflexible architectures did not believe they were inflexible at design time; they discovered this fact when new business needs arose, and they were unable to address them. This uncertainty regarding future needs results in significant risk for organizations.

Packaged applications enable companies to minimize performance, scalability, and deployability risks that arise out of uncertain future business needs. Application vendors remain on the cutting edge of evolving technologies and business needs through their broad customer bases and are far better equipped than any one organization to anticipate, and prepare for, future market needs. Their applications are designed to scale and perform for the largest enterprise customers and offer market-proven architectures and data models.

Companies must recognize that organizations differ in their ability to internally develop DQ applications. Companies with an immature application development organization run a greater risk of a high TCO and a low ROI than those with a mature application development organization: structured organizations (in terms of their use of project management, development methodologies, metrics, data administration, and other factors) tend to produce lower-quality systems. Enterprises poor in application development technology have trouble delivering applications in a timely manner. Enterprises should do a self-assessment to determine their level of application development maturity. This level is likely to be rated lower in light of complex architectures and technologies, such as those found in DQ projects – how to do profiling, matching, standardization, reference data and on and on.

Packaged applications greatly reduce risk because they require much less of the organizations that purchase them. Rather than being able to develop a comprehensive DQ application, organizations must simply be able to maintain a prebuilt application—a task that is orders of magnitude easier.

Organizations that internally develop DQ applications also run the risk that the project's initial scope will increase after the design stage to incorporate additional functionality. This common phenomenon, known as "scope creep," jeopardizes an internal development project because the addition of features and functionality often impairs the original design, which was considered optimal for the functionality then under consideration. If the user experience suffers, then user acceptance may ultimately suffer as well. Scope creep can also result in a delayed deployment, which, in turn, results in additional financial and opportunity costs. Packaged applications, in contrast, do not pose the risk of development scope creep because all the functionality is prebuilt and ready for implementation.

Conclusion

A threshold issue for organizations that execute a DQ strategy is whether to buy or build the enabling technology. Although the build vs. buy decision must be made early in the DQ application life cycle, it has perhaps the greatest potential to impact the required personnel, project budget, and ultimate financial success of the DQ application implemented. Recent surveys suggest that the overwhelming majority of organizations—approximately 75 percent— would prefer to buy packaged DQ applications rather than internally develop applications

ROI analysis can play a critical role in the decision to implement a packaged application. Internally developed DQ applications often suffer from higher ownership costs due to development, maintenance, and opportunity costs. Companies sometimes opt for internal development, hoping that they can obtain the required functionality at a lower TCO, but many experience budget overruns because they underestimate the quantity and quality of the resources required. These companies often find that the TCO for internally developed DQ applications is ultimately higher, not lower, than that for packaged applications. Moreover, internally developed applications often pose greater risk and offer less functionality than their packaged counterparts. Collectively, the higher cost, greater risk, and reduced functionality associated with internally developed solutions can deal a direct and deadly blow to the application's potential benefits and ROI.

Given the substantial costs and risks of internally developed DQ applications, most organizations conclude that packaged applications are the most logical and prudent means by which to execute a DQ strategy. By offering greater functionality, minimizing risk, shortening the time to market, reducing total ownership costs, and increasing ROI, packaged applications enable organizations to maximize shareholder value while improving the overall customer experience. Indeed, packaged applications, by offering less risk and greater rewards than internally developed applications, are a well-carved exception to the rule that rewards follow risk.

EOF

Title: Build or Buy?
Author: William Indest
Managers: Brian Nault and Todd Lesser

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com