

Software Acquisition: The Custom/Package and Insource/Outsource Dimensions

PAUL NELSON
ABRAHAM SEIDMANN

*William E. Simon Graduate School of Business Administration
University of Rochester
Rochester, NY*

WILLIAM RICHMOND

*Perot Systems Corporation
Vienna, VA*

Abstract

Companies have been outsourcing some or all their information systems projects over the past four decades. These projects include applications development and the maintenance of commercial packages or in-house systems. This paper presents a decision framework that captures the major tradeoffs a firm faces when a software acquisition decision is made. This framework and the method of empirical analysis differ significantly from previous work on software acquisition. The software acquisition problem is depicted as two-dimensional, with the firm deciding whether to custom develop the software or base it on a package and whether to insource acquisition tasks or outsource them. Multinomial logit analysis of extensive field data on actual business decisions (not stated rationales) identifies and measures the key factors affecting these two decisions.

We present evidence that a significant interaction exists between decisions on the two dimensions of the software acquisition problem. This "confounding effect" leads firms to make the custom/package and insource/outsource decisions simultaneously. Software acquisition decisions are strongly affected by application properties, technological characteristics and organizational considerations. Surprisingly, we find the software acquisition decision to be largely unaffected by whether or not the system is strategic. Finally, while there is support for the popular belief that software development outsourcing has increased over time, we do not find evidence of such a trend for packages. The framework and empirical results in this paper offer managers a basis for structuring and benchmarking their software acquisition decisions. The paper also characterizes the types of software development and integration projects that it will be most beneficial for vendors to target.

1. Introduction	342
2. The Software Acquisition Cost–Benefit Framework	345
2.1 Cost–Benefit Drivers and the Software Acquisition Decision	346
2.2 System Value (Benefits)	347
2.3 Needs Analysis Costs	347
2.4 Coding and Installation Costs	348
2.5 Monitoring Costs	348
2.6 Contracting Costs	349
3. Hypotheses	349
3.1 Technological Features	350
3.2 Application Properties	350
3.3 Organizational Considerations	352
3.4 Date of Installation	353
3.5 Interaction between the Custom/Package and Insource/Outsource Decisions	353
4. Alternative Models of the Software Acquisition Problem	354
5. Data	357
6. Analysis and Results	358
6.1 Custom/Package and Insource/Outsource Decision Interaction	359
6.2 Technological Features	360
6.3 Application Properties	362
6.4 Organizational Considerations	363
6.5 Temporal Effects	364
7. Conclusions	364
References and Further Reading	365

1. Introduction

Software acquisition is a critical concern to firms worldwide, and the magnitude of its impact on the inner workings of firms is expected to grow. Similarly, the number of firms providing either custom or packaged software services is expected to continue its climb. US firms spend more than \$250 billion annually acquiring software as they automate more functions and try to keep pace with changing technological and business environments (Standish Group, 1994). When these investments in software are successfully implemented, they result in substantial productivity gains and strategic advantages (Neumann, Ahituv and Zviran, 1992; Brynjolfsson, 1993). To realize these benefits, one must identify and understand the considerations that have the most impact on the firm's software acquisition decisions. The cost–benefit framework developed in this paper allows for this investigation, and our empirical analysis provides significant support for the framework.

This paper develops and empirically validates the tradeoffs firms face when they decide on how to acquire a software system. The previous software acquisition literature focuses on the project by project insource/outsource decisions made by firms and typically looks at a single particular aspect of these decisions, such as contracting costs. It is clear that outsourcing, or the purchase of a

packaged solution, does not mean a transfer of project responsibility. Certain tasks can be delegated, but the ultimate responsibility for, as well as the costs and benefits of, the choice of a particular development path reside with the management of that organization. This paper extends the literature through development of a general, economics-based, cost-benefit framework that encompasses both the custom/package and insource/outsource dimensions of the software acquisition problem. Six hypotheses are generated concerning the expected effects of key system characteristics on the software acquisition decision. Logit analysis of actual business decisions is used to test these hypotheses.

The literature has many studies dealing with the outsourcing of data center operations, communications, and users support activities (Hirscheim and Lacity, 1993). The most popular reasons cited for outsourcing these activities include scale economies, improved service quality, price predictability, flexibility, transforming fixed assets costs to variable costs, and freeing human resources. On the other hand, these empirical studies have shown that some benefits are not shared by all users outsourcing their services (Loh and Venkatraman, 1995). Technological dependence on a single vendor could eventually result in higher costs and limited functionalities. Users critically depend upon the economic viability of the service provider and run the risk of losing proprietary information when outsiders deal with their databases. These issues, while important, are not directly relevant to the decision process associated with software acquisition.

Software acquisition projects involve a variety of activities with informational attributes that make it particularly hard to manage. They are not as repetitive as the daily activities involved in running a corporate data center and have major risks and uncertainties that make it difficult for the system to achieve its goals (Banker and Slaughter, 1997; Cusumano and Selby, 1997; Wang, Barron and Seidmann, 1997). The four main issues involve:

- (1) *Information asymmetries.* Users have clear ideas about the potential value of the new system, while developers have a better estimate of the anticipated development costs. As the sophistication of systems grows, they produce fewer tangible and more intangible benefits. By definition, there are no objective economic tools for pricing those intangible benefits.
- (2) *Relationship-specific investments.* Both parties have to spend a significant amount of time communicating the users' priorities and the technological tradeoffs faced by the developers. A common problem is the difficulty in establishing information requirements, both for individual users and for the organization as a whole. These requirements may be too complex, fuzzy, or subject to numerous changes during the course of the project.
- (3) *Lack of market prices.* Applications vary a lot from site to site, making it hard to use market prices as a metric for estimating the true development or integration costs or benefits associated with each particular functionality.

Especially in larger projects, it is hard to estimate the integration needs and the overall cost and time to deliver.

- (4) *Limited observability.* The abstract nature of the final product makes it extremely difficult for users to observe the actual development rate and the overall long-term quality of the system. For instance, it may not be apparent right away to the users that the developer has staffed the project with inexperienced people, and when an external developer runs behind schedule the problem may be hidden for a long time (Mehler, 1991).

Understanding the managerial impact of these issues requires a cost-benefit framework that unifies the important components of the software acquisition problem that have been investigated separately in the literature. It builds upon the contracting and monitoring cost issues addressed in the transaction cost (Coase, 1937; Williamson, 1985, 1989) and incomplete contracting (Grossman and Hart, 1986; Holmstrom and Tirole, 1989; Whang, 1992; Richmond, Seidmann and Whinston, 1992; Richmond and Seidmann, 1993; Chaudhury, Nam and Rao, 1995) literatures and includes the tradeoff between monitoring and production costs developed in the agency theory literature (Ross, 1973; Jensen and Meckling, 1976; Gurbaxani and Kemerer, 1989).

The framework also encompasses the various rationales provided in the empirical literature regarding firms' insource/outsource decisions. These papers report reasons given by managers when asked to explain their decision to either insource or outsource certain information technology (IT) services (i.e., facilities management services). The custom/package decision has not received attention. The primary reasons given for outsourcing include acquiring technical expertise, lowering staff and production costs, and speeding system installation (Altinkemer, Chaturvedi and Gulati, 1994; Meyer, 1994; McFarlan and Nolan, 1995). The primary reasons given for insourcing are that the system requires extensive knowledge of the business or pertains to a core competence, and that contracting costs for outsourcing are high (Lacity, 1992; Jones, 1994; King, 1994; Lacity, Hirscheim and Willcocks, 1994). These same justifications for insourcing or outsourcing are given in the trade press (Caldwell 1989; Gilliam 1990; Harrar, 1993; Evans, 1994). The trade literature cites failure stories involving in-house systems (Groenfeldt, 1997), packages (King, 1997), and outsourced projects (Bulkeley, 1996). Patane and Jurison (1994) investigate software system development in particular and their findings mirror those for IT services. Most of these studies use survey instruments that directly capture the decision rationales from respondents. However, a substantial literature in marketing shows direct subjective measurements of the importance of decision factors provides inferior explanatory power relative to importances inferred from analyzing stated preference or actual decision data (Hauser and Urban, 1977). Our empirical analysis, therefore, extends

the previous literature in that the key factors behind the software acquisition decision are determined from firm actions and not by post-decision rationalizations.

The paper is organized as follows.¹ First, a general, cost-benefit decision framework for the software acquisition problem is developed. We examine the drivers behind this framework and tie these drivers to the underlying technological, application and organizational characteristics of a software acquisition project. Next, we generate six hypotheses concerning the effects of these characteristics on the software acquisition decision. These hypotheses are then investigated using logit analysis of actual choice data. We close with a summary that highlights our empirical findings.

2. The Software Acquisition Cost-Benefit Framework

The software acquisition problem involves two decisions—what acquisition approach to use (whether to custom develop the software or base it on a package) and who should carry out this task (internal resources or external providers). The insource/outsource dimension is analogous to the make/buy decision faced by a manufacturer when it decides who is to make an input. The second decision concerning how the input is to be made (the custom/package dimension), however, makes the software acquisition problem fundamentally different. This two-dimensional, four-option decision is depicted in Fig. 1. The process for using

		Acquisition Team	
		Insource	Outsource
Acquisition Approach	Custom	Internal resources only for needs analysis, coding, etc	Vendor performs needs analysis, coding, etc.
	Package	Internal resources only for package selection, modification, etc.	Vendor performs package selection, modification, etc.

FIG. 1. The software acquisition problem.

¹ An abridged version of this research was previously published as Two dimensions of software acquisition, *Communications of the ACM*, 39, 29-35, 1996.

packaged software involves identifying the company's information processing requirements, obtaining and evaluating information on alternative software packages, selecting a package, possibly modifying the package, and installing and testing the selected system. If an external provider is used to carry out these functions, the acquisition is termed an outsourced package (or {outsource, package}). If in-house staff are used, the acquisition is an insourced package ({insource, package}). When the unique requirements of a user can not be cost-beneficially provided through modifications to packaged software, customized software is developed. Custom development of software generally follows the systems development life cycle or is based on prototyping. These involve a series of possibly repeated steps starting with needs analysis, system design, coding, and testing, and ending with training and installation. An outsourced custom project, {outsource, custom}, uses a vendor to carry out these activities, while an insourced custom development, {insource, custom}, uses only internal resources.

In making the custom/package and the insource/outsource decisions, the company needs to understand the costs and benefits associated with each choice, as well as any interactions between these choices. Typically, the company's goal is to maximize the net present value of the software acquisition, subject to any organizational biases and constraints. Consequently, to make a software acquisition decision properly, the firm must understand the expected benefits of the software system (*system value*) as well as a variety of costs incurred during the acquisition of the system. These include production costs ([a] *needs analysis* and [b] *coding and installation*), and costs related to *monitoring* the acquisition process and to *contracting*.

2.1 Cost-Benefit Drivers and the Software Acquisition Decision

Typically, the financial magnitudes of these five cost-benefit drivers (system value and costs related to needs analysis, coding and installation, monitoring and contracting) differ depending upon whether an internal or external software acquisition team is used (i.e., whether the acquisition is insourced or outsourced). Similarly, these magnitudes differ depending on whether the software is based on a package or custom developed. Table I presents the five cost-benefit drivers and identifies which drivers typically favor a particular acquisition team or approach. Upward pointed arrows (\uparrow) denote which cost-benefit drivers typically favor insourcing over outsourcing and vice versa and which favor custom development over package-based development and vice versa. For example, Table I shows that contracting costs tend to favor insourcing and packages since this cost is typically lower when internal rather than external resources are used to acquire a software system and when the system is based on a package rather than custom developed.

TABLE I
IMPACT OF COST-BENEFIT DRIVERS ON THE SOFTWARE ACQUISITION
DECISION

Cost-benefit driver	Acquisition team		Acquisition approach	
	Insource	Outsource	Custom	Package
System value			↑	
Needs analysis	↑			
Code and install		↑		↑
Monitoring		↑		↑
Contracting	↑			↑

Upward pointed arrows (↑) denote which cost-benefit drivers typically favor different acquisition teams and/or approaches.

The relationships in Table I are important to the system acquisition problem because they identify which cost-benefit drivers typically favor which acquisition options. For example, the decision to insource or outsource depends on (a) the relative sizes of the advantages that insourcing has with respect to the costs of contracting and needs analysis and (b) the relative sizes of the advantages that outsourcing has with respect to monitoring and coding and installation costs. If insourcing's advantage on, say, contracting costs increases, insourcing becomes relatively more appealing.

2.2 System Value (Benefits)

The first of the five cost-benefit drivers presented in Table I is system value. A particular software system is designed to meet possibly unique requirements that support a set of business functions. The value of the system to the firm depends upon the financial importance of these business functions, how effectively and efficiently the system performs the associated requirements and how quickly the software is delivered. In comparing internal to external development of a particular project, system value does not differ because both development teams are expected to deliver the same system. System value is expected to be greater for a custom system since a custom system can be specified so that it better fits the application requirements and thus captures more of the application's potential value. Alternatively, a package is designed to be more general and address requirements common to many firms. Hence, it is less effective in covering a company's unique requirements.

2.3 Needs Analysis Costs

Production costs include costs associated with (a) needs analysis and (b) coding

and installation. Needs analysis involves determining the requirements necessary to automate the desired business application (e.g., customer reservations, manufacturing resource planning and accounts receivable). This requires firm-specific human and informational capital. Needs analysis costs are lower internally since the internal information systems (IS) group better understands the firm's specific business operations and procedures. On the custom/package dimension there are no differences in needs analysis costs because the same needs analysis must be conducted whether the software is acquired as a package or custom developed.

2.4 Coding and Installation Costs

Coding and installation (code and install) includes the detailed software design that translates the users' requirements into a programmable solution, the coding, testing and installation of the programs, and finally the conversion of business operations over to the new system. For software packages, the design, coding and some of the testing costs are included in the purchase price or licensing fee. The actual implementation of the automated business process depends on the firm's existing infrastructure, which is a corporate constraint. Coding and installation costs are higher when the software is developed internally since the external market has more competitive sources of labor and technical expertise, and it is in a better position to take advantage of scale and scope economies (Jones, 1994). As an example, the advent of overseas programming houses in low-wage countries has dramatically driven down prices charged by external developers (Patane and Jurison, 1994). Similarly, due to the scale and scope economies available to vendors, coding and installation costs are lower for packaged software than for custom developed software.

2.5 Monitoring Costs

Software differs from most manufacturing inputs or "physical goods" in that its quality is difficult to assess prior to extended experience with the "product." Economists would refer to software as a credence good. For example, the attribute levels (e.g., size, shape and weight) of most physical goods generally can be ascertained prior to or on the date of delivery through past experience or search. On the other hand, measurement of software system attributes (e.g., functionality, data integrity, response time and the ability to be modified) requires repeated experience with the completed system. Similarly, only imprecise evidence of production progress (the acquisition team's effort and quality) is available since physical input measures such as lines of code or the number of function points are indirect measures, at best, of development progress and system value. These characteristics of software acquisition heighten concerns over opportunistic behavior by employees or vendors that may result in higher coding and installation costs or

lower system value. Monitoring and contracting act as controls on these concerns. Monitoring entails supervising the development team to ensure timely and efficient completion of the system with the desired quality. This includes the costs of managing the project as well as any costs incurred for quality assurance, progress reviews and training. Monitoring costs are lower if external development is undertaken. Software vendors, largely due to scale and scope economies, are typically more efficient at managing software production activities due to better defect prevention and process change management (Joch, 1995). Monitoring costs are lower for packaged software than they are for custom developed systems because of the "physical good" aspects of a package.

2.6 Contracting Costs

Contracting costs include the costs of searching for and evaluating potential custom or packaged software vendors, benchmarking and screening their capabilities, specifying the legal terms of a contract, negotiating the contract's financial details and dispute resolution. In addition, it includes the expected cost of any opportunistic behavior and any risks associated with the possible loss of proprietary information. A contract acts as a legal means to ensure the desired levels of development progress and completed quality. Contracting costs are lower if development is done internally since the company does not have to search for a vendor, negotiate a project-specific contract or use external entities to enforce this contract. Contract disputes over the failure of computer hardware or software to perform as expected are common (Pollack, 1990). Typically, the firm relies on monitoring rather than project-specific contracts for its internal personnel. As with monitoring, contracting costs are lower for packaged software because of the "physical good" aspects of a package.

3. Hypotheses

For a particular software acquisition problem, if the financial values for the five cost-benefit drivers—needs analysis, coding and installation, monitoring, contracting and system value—are known for each of the development options considered, the optimal choice is the option that provides the maximum net present value. Unfortunately, such data are generally not available to managers when making a software acquisition decision because many of the data are typically very difficult to ascertain prior to (or even after) project launch or completion. Prior to an acquisition decision, managers know the characteristics of the desired software and application to be automated, the firm's abilities relative to these characteristics and the organizational attitudes and constraints concerning software development. These determinants of the decision process center on three

factors: technological features, application properties and organizational considerations. The first two factors directly affect the five cost–benefit drivers and, hence, steer the manager’s decisions on both dimensions of the software acquisition problem in predictable ways. Organizational considerations may or may not affect the cost–benefit drivers, but in either case they influence the software acquisition decision.

Table II outlines the intuitive relationships among these three factors, the five cost–benefit drivers and the two dimensions of the software acquisition decision. Four hypotheses discussed below address the expected marginal impacts of technological, application and organizational factors on a particular software acquisition decision. A temporal effect on this decision is also hypothesized because the trade literature frequently indicates that outsourcing has become more popular over time. Finally, we ask whether decisions on these two dimensions of the software acquisition problem interact.

3.1 Technological Features

According to both managers and researchers, one of the primary reasons for outsourcing software development is to obtain access to expertise in specialized technologies and advanced development environments (Meyer, 1994; Patane and Jurison, 1994). External vendors can leverage expertise and design concepts across numerous projects and firms, thereby incurring scale and scope economies that are unlikely to be available to the typical firm. Consequently, as the technology becomes more specialized or advanced, coding and installation costs more strongly favor outsourcing. Since more complex technology is more difficult and technically demanding, it also increases monitoring and contracting costs. However, while the relative increase in the importance of monitoring costs is likely to be smaller than that for contracting costs, outsourcing’s increased coding and installation cost advantage is likely to dominate. As shown in Table II, the increased importances of the costs associated with coding and installation, monitoring and contracting all favor packaged software.

Hypothesis 1: Systems using specialized technology or advanced development environments are, all else equal, more likely to be outsourced and/or associated with packaged software.

3.2 Application Properties

Managers and researchers frequently cite two application properties—the application’s uniqueness and its strategic role—as strongly favoring a decision to insource (Jones, 1994; Lacity, Hirscheim and Willcocks, 1994). Strategic applications are intended to provide a company with a competitive advantage over its

rivals. The intimate relationship between a strategic application and a firm's core competencies results in higher system value. Consequently, needs analysis costs and the costs associated with potential opportunistic behavior (monitoring and contracting costs) become relatively more important. For such an application, the higher contracting costs associated with outsourcing are expected to exceed insourcing's higher monitoring costs. This, coupled with the increased importance of needs analysis, favors the use of internal development. Similarly, the effect on system value is expected to be stronger than the effect on potential opportunistic behavior. Thus, the model predicts that strategic systems are more likely to be custom developed.

Hypothesis 2: Strategic applications are more likely to be insourced and/or custom developed, all else equal.

For applications that are common to many organizations and that use standard data structures, vendors can leverage their investment and labor pool across multiple clients. This effectively increases the relative advantage of outsourcing and packaged software with respect to the coding and installation costs. In a similar vein, a unique system specifically designed and tailored to a particular company's requirements entails higher needs analysis costs, resulting in a tendency toward insourcing.

Hypothesis 3: Common systems are more likely to be outsourced and/or acquired as packaged software, all else equal.

3.3 Organizational Considerations

A company's culture and the system's sponsor have been identified as significant factors in software acquisition decisions (Arnet and Jones, 1994; Nault and Richmond, 1995). Examples of possible organizational biases are the "not-invented-here syndrome" and "kingdom-building" (Jensen and Meckling, 1976). These two particular biases result in a tendency toward insourced, custom development. Alternatively, a firm or system sponsor may have biases that favor outsourcing or packaged software. Organizational biases of this type can be seen as covariates to the cost-benefit model.

Other organizational factors, such as worker empowerment or the ability to retain employees with desirable state-of-the-art technical skills, may affect the relative costs and benefits of packaged software versus custom development and insourcing versus outsourcing. For example, a firm may excel in managing vendor relationships, which reduces the relative importance of contracting costs.

Hypothesis 4: Organizational biases favor insourced and/or custom software acquisition, all else equal.

3.4 Date of Installation

Recent publications propose four reasons for a perceived increase in the level of outsourcing over time: the increased visibility of outsourcing, an increased focus by firms on their core competencies, the increased availability of outside vendors and management blindly following the latest fad (Loh and Venkatraman, 1992).

Two alternative explanations argue against inclusion of temporal effects as an additional covariate to our model. One possibility is that, while the absolute amount of outsourcing has increased due to an increase in the total number of functions being automated, the percentage of projects outsourced has not changed over time. This could also be said for the use of packaged software. A second explanation for increased outsourcing or use of packaged software is that technological features and application properties have changed over time in a manner that favors outsourcing and packages. Both of these alternative explanations imply that the date of system installation has no incremental effect on the software acquisition decision beyond those of the technological, application and organizational factors.

Hypothesis 5: Outsourced software acquisition has become more likely over time, all else equal.

3.5 Interaction between the Custom/Package and Insource/Outsource Decisions

The custom/package and insource/outsource decisions are related since they share the same five cost-benefit drivers. If, in addition, these two decisions interact, software acquisition is fundamentally different from a manufacturer's unidimensional make/buy decision. That is, the "confounding effects" of each decision on the other lead the firm to make these decisions in combination, rather than separately. The use of packaged software lowers the relative importance (monetary value) of all the drivers except needs analysis costs, with coding and installation costs the most affected. Since both of the drivers that favor outsourcing—coding and installation costs and monitoring costs—are negatively affected, the use of a package favors insourcing. Analogously, custom development favors outsourcing. Alternatively, outsourcing reduces the importance of coding and installation costs and increases the importance of contracting and monitoring costs. Insourcing leads to the opposite effects. Neither leads to a clear-cut custom or packaged development tendency. These interactions increase the complexity of the software acquisition decision. Correspondingly, this results in either a sequential decision structure, with the custom/package decision preceding the insource/outsource decision, or a simultaneous choice among four acquisition options—{insource, package}, {insource, custom}, {outsource, package} and {outsource, custom}.

Hypothesis 6: An interaction exists between the insource/outsourcing and custom/package decisions.

4. Alternative Models of the Software Acquisition Problem

We have constructed three models to identify the existence and type of interaction between the custom/package and insource/outsourcing decisions. Underlying each of the three hypothesized models is the idea that the utility of a software acquisition option depends on its perceived costs and benefits, which in turn depend on the system's technological, application, organizational and temporal factors. This is similar to the traditional multiattribute utility framework used in marketing and economics to model brand choice or preference with respect to a set of products or services (Lancaster, 1966; Rosen, 1974; Green and Srinivasan, 1990). For any particular project m , the utility of acquisition option j is

$$U_j = \alpha_j + \sum_i \beta_{ij} X_{im}. \quad (1)$$

The α_j intercept term is an option-specific constant depicting a firm's predisposition to use option j (i.e., organizational biases or corporate policy). X_{im} represents the level of explanatory variable i intrinsic to project m . These explanatory variables measure the technological, application, organizational and temporal factors. If the explanatory variable i is nominal-scaled, X_{im} is a binary (dummy) variable equal to one if variable i is present in project m and zero otherwise. For example, a possible explanatory variable—let us call it variable 1—may pertain to whether or not a third-generation programming language is used, resulting in X_{1m} equal to one if the system incorporates a third-generation language and zero otherwise. Similarly, explanatory variable 2, X_{2m} , may equal one if the application is strategic and zero otherwise. The β_{ij} terms reflect the impact of a particular explanatory variable i on the utility of option j . Note that, following the discussion in the hypotheses section, the explanatory variables may have differential effects on the utility of each acquisition option (i.e., β_{ij} rather than β_i is modeled).

The firm wishes to choose the acquisition option with the largest utility. Hence, for a particular project the probability that the firm chooses option j over option k is

$$P[j \text{ preferred to } k] = P[U_j > U_k]. \quad (2)$$

It follows that the probability that option j is preferred to all other acquisition options is

$$P[j] = P[U_j > U_k: \text{ for every } k, k = 1, 2, \dots, K, k \neq j]. \quad (3)$$

If the utilities can be calculated with certainty, these probabilities equal either zero or one. However, if uncertainty exists due to unknown or unmodeled factors, the estimate of each acquisition option's utility equals its true utility plus an error

term, $U_j^{est} = U_j + \varepsilon_j$. This results in

$$P[j] = P[U_j^{est} - U_k^{est} > \varepsilon_k - \varepsilon_j; \text{ for every } k, k = 1, 2, \dots, K, k \neq j]. \quad (4)$$

The error terms are commonly assumed to be independently distributed random variables with a Gumbel distribution. This assumption results in the multinomial logit model, which has an intuitively appealing representation for the probability of choice of each possible option (McFadden, 1974; Gaudagni and Little, 1983). The probability that option j is chosen depends on the ratio of an exponential transformation of option j 's utility to the sum of exponential utility transformations over all options including j :

$$P[j] = \frac{e^{U_j^{est}}}{\sum_{k=1}^K e^{U_k^{est}}}. \quad (5)$$

For expository ease and identification reasons, we define option 1 as a base case with utility normalized to zero. For a particular project, the "utilities" of the other options then depict the differences in utility between each particular option and the base case (option 1). Mathematically, for project m , the probability that option j is chosen is

$$P[j] = \frac{e^{U_j^{est}}}{\sum_{k=1}^K e^{U_k^{est}}} = \frac{e^{\{\alpha_j + \sum_i \beta_{ij} X_{im}\}}}{\sum_{k=1}^K e^{\{\alpha_k + \sum_i \beta_{ik} X_{im}\}}} \quad (6)$$

$$= \begin{cases} \frac{1}{1 + \sum_{k=2}^K e^{(\alpha_k - \alpha_1) + \sum_i (\beta_{ik} - \beta_{i1}) X_{im}}} & \text{for option 1 } (k = 1) \\ \frac{e^{(\alpha_j - \alpha_1) + \sum_i (\beta_{ij} - \beta_{i1}) X_{im}}}{1 + \sum_{k=2}^K e^{(\alpha_k - \alpha_1) + \sum_i (\beta_{ik} - \beta_{i1}) X_{im}}} & \text{for options } j = 2, 3, \dots, K. \end{cases} \quad (7)$$

Setting $\gamma_j = \alpha_j - \alpha_1$ and $\delta_{ij} = \beta_{ij} - \beta_{i1}$, these equations simplify to

$$P[1] = \frac{1}{1 + \sum_{k=2}^K e^{\gamma_k + \sum_i \delta_{ik} X_{im}}} \quad (8)$$

$$P[j] = \frac{e^{\gamma_j + \sum_i \delta_{ij} X_{im}}}{1 + \sum_{k=2}^K e^{\gamma_k + \sum_i \delta_{ik} X_{im}}}, \quad j = 2, 3, \dots, K. \quad (9)$$

The higher the utility of option j relative to that of the base case, the greater is its probability of being chosen. It follows that technological, application, organizational and temporal factors that favor option j relative to the base case have positive parameter values for the γ_j and δ_{ij} terms, and those that favor the base case have negative values. So, if for option j the variable measuring whether or not the application is strategic (called X_{2m} earlier) is found to have a positive parameter value δ_{2j} , the utility of option j is δ_{2j} greater than the utility of option 1, the base case. Given equations (8) and (9), this means that the probability of choice for option j is higher if the application is strategic, while for option 1 this probability is lower.

The first possible model of software acquisition is a simple four-option version of the multinomial logit model proposed in equations (8) and (9). This model assesses the probability of each of the four {insource/outsourcing, custom/package} combinations, $P[\text{insource/outsourcing, custom/package}]$, directly. The base case option 1 is defined as {insource, custom}. Option 2 is {outsourcing, custom}; option 3 is {insource, package}; option 4 is {outsourcing, package}.

Our second model of software acquisition behavior assumes that decisions on the two dimensions of the software acquisition problem do not interact. This assumption simplifies the representation of the firm's decision to the product of two binomial (two-option multinomial) logit models:

$$P[\text{insource/outsourcing, custom/package}] = P[\text{insource/outsourcing}]P[\text{custom/package}] \quad (10)$$

$$P[\text{insource}] = \frac{1}{1 + e^{\gamma^{i/o} + \sum_i \delta_i^{i/o} X_{im}}}; \quad P[\text{outsourcing}] = \frac{e^{\gamma^{o/o} + \sum_i \delta_i^{o/o} X_{im}}}{1 + e^{\gamma^{i/o} + \sum_i \delta_i^{i/o} X_{im}}} \quad (11)$$

$$P[\text{custom}] = \frac{1}{1 + e^{\gamma^{c/p} + \sum_i \delta_i^{c/p} X_{im}}}; \quad P[\text{package}] = \frac{e^{\gamma^{c/p} + \sum_i \delta_i^{c/p} X_{im}}}{1 + e^{\gamma^{c/p} + \sum_i \delta_i^{c/p} X_{im}}} \quad (12)$$

The third hypothesized model assumes a particular type of interaction exists between the custom/package and insource/outsourcing decisions. It is motivated by a few managers who suggested that their firm first decides whether or not to use a package and then decides whether to outsource or insource. This model assumes a nested decision process in which the insource/outsourcing decision is conditional on the custom/package decision. A nested logit framework results in which $P[\text{insource/outsourcing, custom/package}]$ can again be represented as the product of two binomial logit models (McFadden, 1981; Ben-Akiva and Lerman, 1985):

$$P[\text{insource/outsourcing, custom/package}] = P[\text{custom/package}]P[\text{insource/outsourcing given custom/package}] \quad (13)$$

$$P\left[\text{insource}|\text{custom}\right] = \frac{1}{1 + e^{\gamma^c + \sum_i \delta_i^c X_{im}}};$$

$$P\left[\text{outsource}|\text{custom}\right] = \frac{e^{\gamma^c + \sum_i \delta_i^c X_{im}}}{1 + e^{\gamma^c + \sum_i \delta_i^c X_{im}}} \quad (14)$$

$$P\left[\text{insource}|\text{package}\right] = \frac{1}{1 + e^{\gamma^p + \sum_i \delta_i^p X_{im}}};$$

$$P\left[\text{outsource}|\text{package}\right] = \frac{e^{\gamma^p + \sum_i \delta_i^p X_{im}}}{1 + e^{\gamma^p + \sum_i \delta_i^p X_{im}}} \quad (15)$$

$$P[\text{custom}] = \frac{1}{1 + e^{\gamma^* + \sum_i \delta_i^* X_{im} + \phi_p CV_p - \phi_c CV_c}} \quad (16)$$

$$P[\text{package}] = \frac{e^{\gamma^* + \sum_i \delta_i^* X_{im} + \phi_p CV_p - \phi_c CV_c}}{1 + e^{\gamma^* + \sum_i \delta_i^* X_{im} + \phi_p CV_p - \phi_c CV_c}}$$

$$CV_p = \ln[1 - e^{\gamma^p + \sum_i \delta_i^p X_{im}}] \quad (17)$$

$$CV_c = \ln[1 - e^{\gamma^c + \sum_i \delta_i^c X_{im}}]. \quad (18)$$

5. Data

Survey data were collected from five companies concerning every documented software development project that each firm had ever undertaken. Each firm's actions (revealed preferences) rather than post-decision rationalizations for each custom/package, insource/outsource decision pair were ascertained. This allows our analysis to differ from and improve upon previous studies by investigating the factors affecting what firms do rather than what firms say affects their actions. In a similar situation, marketers find estimations of attribute importances based on stated brand preferences or choice to provide superior explanatory power relative to the direct statements of attribute importances from respondents (Hauser and Urban, 1977).

Multiple measures of the technological, application and organizational factors were collected, as was the date of each system's installation. The data cover 186 system projects ranging from small personal computer applications to large distributed systems and from common accounting systems to strategic operational systems. System installation dates ranged from 1967 to 1993. A single individual

in each company filled out all of the firm's surveys, which were then reviewed with the authors. The participating companies range in size from \$20 million to \$900 million in annual revenue. They include a regulated utility, a long-distance telephone company, a payroll processing company, a rocket engine manufacturer and a building products manufacturer. Although the sample may not be representative of industry in general, it does include a broad cross-section of applications, industries and company sizes.

The two dimensions of the software acquisition decision for each project were recorded. Sixty-four percent of projects were {insource, custom}. Seventeen percent were {outsource, custom}. Eleven percent were {insource, package}, and the remainder (eight percent) were {outsource, package}.

Five technological features for each system were specified by the survey respondent:

- (1) database management system (DBMS);
- (2) programming language;
- (3) hardware platform;
- (4) system architecture;
- (5) processing mode.

Information regarding the operating system(s) and on-line monitor(s) was also collected. Since these responses were highly correlated with the hardware platform responses, they were not analyzed further.

Four variables were ascertained to measure each system's application properties. The first three measures pertain to the application's uniqueness, while the fourth pertains to its strategic role:

- (1) application type (organizational level supported);
- (2) functional area;
- (3) application uniqueness (relative to other firms in the industry);
- (4) strategic mission.

Two variables measure organizational considerations:

1. system sponsor (organizational level);
2. firm (binary variables used to encompass idiosyncratic organizational biases and constraints).

In order to measure a possible time trend, the installation date of the system was recorded.

6. Analysis and Results

We begin by determining the most appropriate logit model and corresponding view of the software acquisition problem. For the chosen model, we then discuss the

relative importances of the technological, application and organizational factors to the software acquisition problem. Finally, we address a possible temporal effect.

6.1 Custom/Package and Insource/Outsource Decision Interaction

For each of our 186 observations, the variables measuring the technological, application, organizational and temporal factors are nominal scaled. Hence, each variable i is represented by a binary (dummy) variable, X_{im} . For each of the three models developed, these binary variables act as the right-hand-side independent variables in the non-linear probability equations outlined in Section 4.

The dependent variable in the three models is the insource/outsource, custom/package decision. The nominal scale of these data rules out the use of regression analysis as a tool to estimate the relationship between the independent variables and the insource/outsource, custom/package decision. Consequently, a maximum likelihood procedure is used to estimate each logit model's parameters (the γ 's and δ 's). This complex procedure basically computes the set of parameter values that at the aggregate level maximizes for each project (observation) the estimated probability of choice for the option actually chosen, and minimizes the estimated choice probabilities for all the other options. For mathematical details see McFadden (1981).

Statistical evaluation is used to determine which of the three decision models presented in Section 4 best depicts firm behavior. Since the model assuming that the custom/package and insource/outsource decisions do not interact is not a restricted version of the four-option multinomial logit model, we use Akaike's information criteria (Akaike, 1974) to compare the models. This criterion is analogous to the comparison of adjusted r -squares in regression analysis and strongly supports the four-option multinomial logit model. A nonparametric χ^2 test was also carried out concerning the hypothesis that $P[\text{insource/outsource, custom/package}] = P[\text{insource/outsource}]P[\text{custom/package}]$. This test also rejected the hypothesis of no interaction at the 5% level. Consequently, we find support for the hypothesis (Hypothesis 6) that an interaction exists between the insource/outsource and custom/package decisions.

The data do not support the notion that the insource/outsource decision is conditional on the custom/package decision. Since the four-option multinomial logit model is a restricted version of the nested logit model depicted in equations (13)–(18), a comparison of the two models' log likelihood values is applicable. For the nested logit model to be appropriate, the category value parameters, ϕ_c and ϕ_p , must be statistically significantly different from zero; otherwise, the four-option multinomial logit model results. These conditions are not borne out in the estimation results. The hypothesis that ϕ_c and ϕ_p equal zero can not be rejected at even the 10% level.

Based on these findings, the four-option multinomial logit model expressed in equations (8) and (9) best depicts firm behavior and is used to investigate Hypotheses 1 through 5. The parameter estimates for this four-option multinomial logit model, the γ_j s and δ_{ij} s in equations (8) and (9), are provided in Table III. In general, they show significant support for our hypotheses. Note that some explanatory variables for particular options have been dropped in order to simplify the analysis, reduce collinearity problems and focus attention on the key decision factors. If a variable was dropped, it was not statistically significant (one-sided) at the 10% level. Also note that a positive parameter value means that the probability of the denoted acquisition option increases relative to the base case {insource, custom}, holding all else constant. A negative parameter value means that the probability of the denoted acquisition option decreases relative to {insource, custom}, holding all else constant. For example, Table III presents a positive {outsource, custom} parameter value (22.80) for the dummy variable equal to one for Firm 2 and zero otherwise. This implies that, holding the effects of all other explanatory variables constant, Firm 2 is more disposed toward outsourced custom development than insourced custom development (the base case). Moreover, since this Firm 2 parameter is larger than both the Firm 2 parameter for {insource, package} and the Firm 2 parameter for {outsource, package} (4.28 and 5.63, respectively), Firm 2 is more likely to use outsourced custom development than any of the other three acquisition options, holding the effects of all other explanatory variables constant. The discussion of each parameter estimate that follows implicitly holds all else constant.

6.2 Technological Features

We find general support for the idea that systems using specialized technology or advanced development environments are more likely to be outsourced and/or associated with packaged software (Hypothesis 1). Estimated parameter values pertaining to four of the five technological factors provide support for this hypothesis.

- (1) *Database management system (DBMS)*. We find that the {outsource, package} option is less likely when no database system (i.e., a file-based system) is used. This follows our hypothesis, since file-based systems use the simplest, most standard technology and are not associated with advanced development environments.
- (2) *Programming language*. We find that systems written in third-generation languages are less likely to be {outsource, custom}, and those written in fourth-generation languages are more likely to be {outsource, package}. These findings support Hypothesis 1, since third-generation languages are a standard technology, whereas fourth-generation languages are more specialized and are associated with advanced development environments.

TABLE III

FOUR-OPTION MULTINOMIAL LOGIT RESULTS

System characteristics	Variables	{Outsource, custom } parameters	{Insource, package } parameters	{Outsource, package } parameters
Technological features	H₁			
DBMS	File based	—	—	-2.17***
Programming language	Third-generation	-0.70	—	—
	Fourth-generation	—	—	2.18***
Hardware platform	Minicomputer	2.01***	4.34***	3.63***
	Multiple	1.61***	1.81*	2.32**
Architecture	Distributed	—	—	—
Processing mode	Batch	2.22***	—	—
Application properties	H₂, H₃			
Strategic mission	Strategic	—	0.98*	—
Application uniqueness	Common	0.89*	2.44***	2.77***
Application type	Transaction processing	1.43**	—	—
	Combination	1.42*	—	1.61**
Functional area	Finance	—	3.14***	—
Organizational considerations	H₄			
Firm	Firm 2	22.80***	4.28***	5.63***
	Firm 3	—	—	3.88**
	Firm 4	22.80***	4.28***	5.63***
	Firm 5	22.80***	2.61**	5.63***
Sponsor level	Executive	—	—	—
Time of installation	H₅			
	Before 1981	-2.67***	—	—
	After 1990	—	—	—
Intercept term	H₄			
	Intercept	-25.73***	-9.91***	-10.47***
Goodness of fit				
	Log likelihood value	-123.0***		
Significance levels (one-sided)				
	*** 1% level			
	** 5% level			
	* 10% level			

Results pertain to a four-option version of the model expressed by equations (8) and (9). The parameter estimates provided relate to the likelihood of the three noted acquisition options relative to a base case, which is {insource, custom}.

