Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 146 (2003) 339–351
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

A multi-period optimization model for the procurement of component-based enterprise information technologies

R.P. Sundarraj [a,*], Srinivas Talluri [b,1]

[a] Department of Management Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada
[b] Department of Marketing and Supply Chain Management, Eli Broad College of Business, N370 North Business Complex, Michigan State University, East Lansing, MI 48824, USA

Abstract

Today, timely sharing and coordination of information across the supply chain are key factors to improving the performance of an organization. Although single-vendor enterprise information technologies (EITs) are seen as enablers for accomplishing this goal, a number of organizations are concerned about the high cost and time involved in such a solution. An alternative new paradigm has emerged in the form of componentized EITs, which are stand-alone software components that can be easily integrated with one other, and which provide the advantages of easing cash-flow problems and of decreasing customization times. Research in the evaluation of componentized EITs lacks objective assessment tools that simultaneously consider both procurement and inter-component integration issues. This paper fills this gap by proposing a multi-period integer-programming model to assist decision-makers in the procurement of componentized EITs. We consider component costs as well as inter-component integration costs over a given planning horizon and provide managerial insights based on different experiments with the model.

Keywords: Enterprise information technologies; Componentized enterprise technology; Software components; Software frameworks; Integer programming; Procurement; Supply chain management

1. Introduction

There is conflicting evidence about the efficacy of a traditional enterprise information technology (EIT) to an organization. On the one hand, because EITs provide information from the entire supply chain and beyond, they are often seen as enablers of breaking down the walls of managerial functionalism and thereby improving organizational performance. On the other, there have been questions about the bottom-line productivity of information technology (IT) in general (Brynjolfsson, 1993), and of EITs in particular (Cliff, 1999). This skepticism is not surprising considering that the average return on EIT investment is negative, and the average implementation cost and time are $10.3 million and 23 months, respectively (Anonymous, 1999).

* Corresponding author. Address: Department of Management Science, University of Waterloo, 200 University Ave West, Waterloo, Ont., Canada N2L 3G1. Tel.: +1-519-888-4567x2235; fax: +1-519-746-7252.
E-mail addresses: rsundarr@engmail.uwaterloo.ca (R.P. Sundarraj), talluri@pilot.msu.edu (S. Talluri).
[1] Tel.: +1-517-353-6381.

0377-2217/03/$ - see front matter
PII: S0377-2217(02)00553-2

Componentized EITs are aimed at resolving this conflict. Componentization is an evolution from software components used for object-oriented systems (Kumar and Van Hillegersberg, 2000). Components hide their complexities through encapsulation, communicate with one another through interface mechanisms, and are reconfigurable and extendible, as needed. Componentized EITs offer a number of advantages over traditional single-vendor enterprise systems (Fan et al., 2000; Sprott, 2000; Kumar and Van Hillegersberg, 2000). First, firms would be able to incrementally acquire and assemble component-based EIT systems, purchased from multiple vendors and customized to the company's specific needs. Second, componentization should lower the cost of acquiring and implementing reasonably customized EIT systems for small- and medium-size enterprises. Third, their introduction and migration may become less revolutionary and more evolutionary, as outdated components are upgraded instead of the whole system (upgradability, expansion flexibility). Finally, in multi-site implementations, the versions implemented at each site can be tailored to the site itself (adaptability).

However, componentized EITs are just emerging, and optimization models for their evaluation do not exist. The need for this evaluation is critical, especially because IT is undergoing heavy scrutiny from managers and chief executives (Torkzadeh and Doll, 1999). Our paper fills this gap by proposing a multi-period optimization model that considers procurement as well as integration costs. We provide managerial insights based on different experiments with the model. In the next section, we review the basic organizational issues and reasons for the componentization of EIT systems as well as a literature survey of models for evaluating such systems. In Section 3, we provide an optimization model, while Section 4 discusses the implications with the model. Section 5 gives an extension that considers budgetary allocation mechanism, and Section 6 outlines our concluding remarks along with future work.

The contributions of this paper include: (i) the identification of issues related to modularization/componentization of EITs and their evaluation; (ii) an introduction of a new model for this evaluation; and (iii) using the model to obtain insight into managerial issues concerning the evaluation of EITs.

2. Background

First, in Sections 2.1 and 2.2 respectively, we describe the background of traditional and componentized EIT systems. Section 2.3 traces the relationship of componentized systems to emerging concepts in managing the supply chain, while Section 2.4 provides an overview of the existing evaluation methods of IT systems.

2.1. Noncomponentized systems

Organizations have been continually challenged by the issues of how to integrate IT into their business strategy. Due in part to the rapid evolution of IT, a number of existing systems ("legacy" systems) created just a decade ago are now becoming obsolete and encountering numerous difficulties. The term legacy is somewhat broad and includes systems which: (i) lack the efficiency to scale for the volume of today's data sets; (ii) were written for hardware/software that are no longer seen as part of an organization's IT strategy; (iii) fail to interoperate (Alderson and Shah, 1999) with current hardware/software systems; or (iv) were tailored for the business rules of increasingly obsolete organizational forms. Thus, on the one hand, legacy systems are seen as old, inflexible, nonportable and undocumented. On the other, owing to their long periods of existence, they have often become business-critical applications in a number of instances. This situation has led to research to look for ways to integrate legacy systems with current technology (e.g., Burd and Munro, 1998; Markosian et al., 1994).

This integration of legacy systems, however, could be construed only as a temporary measure. A more long-term solution is to reengineer IT along with the business, by employing enterprise

resource planning (ERP) systems. [2] ERP systems did start off as solutions for large Fortune 100 companies, but have been expanding into medium and small sized companies as well (Holt, 1999), thereby causing market research firms to predict an industry-wide annual growth rate of 37% and a revenue of $52 billion by the year 2002 (Trunick, 1999). ERP systems have embedded in them business models of industry's "best practices" (Soh et al., 2000), and therefore, represent a hybrid approach between costly customized software and packaged software that cannot be altered to suit business needs (Sawyer, 2000). Through an integrated support of numerous functions and of geographically dispersed departments, an ERP system makes operations within and between functions seamless.

This feature carries with it a number of strategic and operational advantages, but it could also be the reason for an ERP system to become monolithic. By the manner in which it is designed, an ERP system is a single piece of software embodying the client–server paradigm, with the client managing the user-interface part and the server providing data and imposing business rules embedded in the system. These embedded rules constrain the business process that can be adopted by the organization (Davenport, 1998; Fan et al., 2000; Kremers and Van Dissel, 2000), and because an ERP system is a single piece of software, the rules apply throughout the organization. Thus, a significant portion of an ERP implementation is spent in ironing out the processes that need to be followed by the company as a whole, causing ERP implementations to be extremely costly and time consuming. While the ERP systems can be modified to fit the changing needs of organizations, it is usually a very difficult task to reconfigure them. Thus, if there is a better process to perform a function that is not already coded into an ERP system, it becomes an arduous task to implement that process. Thus, firms may be forced to adopt less effective processes. Moreover, ERP systems are seen as execution systems, with optimized planning and decision-support being provided by other software packages (such as advanced planning and scheduling systems) that "bolt-on" to them. Thus, it may be difficult to plan for an effective enterprise wide solution with existing ERP systems.

The aforementioned difficulties are motivating software engineers to design other types of EITs, by adapting from and expanding upon the field of component-based software development (Boehm, 1999). In the next section, we trace the various issues pertaining to componentized EITs.

2.2. Componentized EITs

2.2.1. Overview of componentized EITs

Component-based software development is changing the way that large software systems are developed (Clements, 1996). This paradigm of software development is based on the premise that there is sufficient amount of commonalty among many large systems of a specific application domain. Thus, by using software components that contain these common functionalities, it should be possible to create a number of complex systems with relative ease (Vitharna and Jain, 2000). This concept has support from the software-engineering literature. For example, Staringer (1994) describes how components helped develop a mission-critical financial-risk-management application both quickly and inexpensively. Likewise, Baumer et al. (1997) describe their experiences in developing components for a heterogeneous group of 450 banks. The family of components covered almost the entire area of banking, and consisted of several frameworks and 2500 classes. Further, to facilitate easy access, components are downlodable from the Internet or from components banks (Jeong et al., 2001).

However, with component systems, the focus of software development has shifted from software implementation to one of "composing" software systems (Clements, 1996). Hence, in contrast with a monolithic system from a single vendor, components from different vendors must be integrated with one another to develop a system. This could

[2] An ERP system, as used in this context, is narrower in focus as compared to EITs. For example, ERP systems such as SAP are primarily execution systems without a significant component of planning modules, whereas by EIT, we mean an agglomeration of interconnected systems that together provide an enterprise-wide solution.

become an issue, if there is mismatch among the components (Garlan et al., 1995). We next discuss the technologies that mitigate integration issues, and thereby provide a net advantage to the paradigm of component-based systems development.

2.2.2. Integration among components

A key issue in componentized systems is the methodology used to integrate the components in question. Components communicate with one another through *interfaces*. An interface is a contract that specifies how a component's functionality is accessed (Stets et al., 1999). Since all components hide how their functionality is implemented, interfaces serve as a major differentiating factors among different types of components. Researchers have developed a reference model for different types of components stemming from differences in their interfaces (Beugnard et al., 1999; Brown and Wallnau, 1996; Brown and Wallnau, 1998).

First, in *components-off-the-shelf* (COTS), the interfaces are restricted to specifying the methods that are provided in order to make a system work. However, in a number of situations one must understand other aspects of a component's performance such as reliability, scalability, usability, quality of the solution, and time to generate a solution (Beugnard et al., 1999). For example, one of the main reasons for the failure of Foxmeyer drug's ERP implementation is that the systems failed to scale to the volume of transactions at the company (Appleton, 1997). Similar problems arise with real-time applications for which a component's mean time between failures and timing performance are attributes that crucially affect the workings of the overall system. This leads us to interfaces of *qualified components* (in the sense of Brown and Wallnau (1996) and Brown and Wallnau (1998)), in which detailed performance indicators, which are rarely given with COTS, are specified.

Finally, we consider *framework-based components*. A framework is a reusable, standard, tailorable software architecture consisting of a collaborating set of components (Demeyer et al., 1997). A framework facilitates in the design and creation of open components and thereby permits flexible integration of components from third-party vendors (Demeyer et al., 1997; Sousa and Garlan, 2001). Usually, a framework defines three things (Sousa and Garlan, 2001): (i) the overall structure of an application in terms of its major types of constituent components; (ii) a set of interface standards that describe the capabilities and performance characteristics of those components; and (iii) a reusable architecture that support the integration of those components. Thus, component mismatch (Garlan et al., 1995), a frequent issue with component integration, is greatly reduced according to Sousa and Garlan (2001). By providing such an integration platform for component developers, application developers can build new systems from a rich collection of choices.

Examples of frameworks include the San Francisco Project from IBM, the Eagle Project from Anderson Consulting and Enterprise JavaBeans (EJB) from Sun Microsystems (Jeong et al., 2001). Frameworks frequently breakdown their architecture into layers. For example, in EJB, there is a *client or presentation* layer, a *business logic* layer and a *server* layer. As shown in Fig. 1, components in one layer of a framework do not directly interact with those of another layer (Sousa and Garlan, 2001). That is, a client invoking a business application first contacts its interface, which then interacts with the container interface and through that with the component itself. This is also the case with how transaction-management and security requests get handled between the business-logic layer and the server layer. Note how this working contrasts with that of monolithic client–server systems (e.g., SAP), in which the software directly provides services to the client programs (e.g., see dotted lines from client to order component and from order component to transaction management). This implies that traditional components will have to deal with the interfaces of all other components with which it might need to interact, as opposed to a limited set of interfaces prescribed with the framework. The next section discusses the implications of such frameworks to EITs.

2.2.3. Advantages of componentized EITs

The framework-based components described above are useful to the development of EITs. A

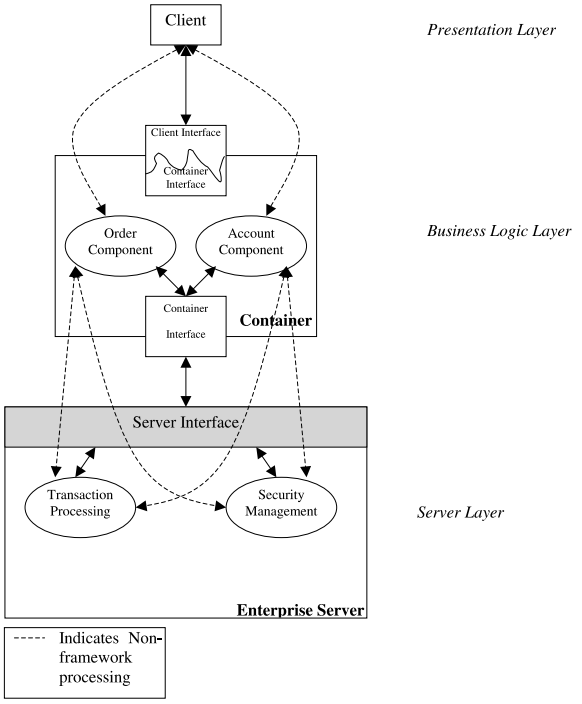

Fig. 1. Illustration of a framework.

coherent set of components can interoperate with one another to achieve a high-level business function (Vitharna and Jain, 2000; Fan et al., 2000). Hence, a number of researchers argue that traditional monolithic mainframe-based business applications are behind us, and that the trend is to use frameworks to build enterprise systems that can operate in a distributed-computing environment (see Orfali et al., 1996; Fingar, 2000; Lunney and McCaughey, 2000).

Componentized EITs developed through the use of such frameworks can be useful for organizations, both small and large. Large organizations do not have to go through the arduous process of reconciling conflicting processes. Instead, different sections of an organization can have different versions of a componentized system, all communicating with one another through the interface mechanisms described above. For small companies, the cost burden of implementing EIT can be lessened, as they can pick and choose which components to implement in their organization. They can hence allow for incremental implementation, [3] and thereby, use pay-as-you-go approaches to help finance these systems (Kumar and Van Hillegersberg, 2000). Also, as new and improved components are developed for performing a particular process (e.g., a component that scales well for high volumes), firms can replace the existing ones for improving performance, thereby lowering implementation cost. Again, such a replacement does not have to be done throughout the organization, but it can be restricted to those divisions/subdivisions that actually need improved performance. In summary, componentized EIT's provide an opportunity for companies to select the best of breed to perform an individual function (Chopra and Meindl, 2001). While integration is a critical issue, the software-engineering literature cited above contends that the advantages outweigh this issue and predict such systems to become prevalent. It is, therefore, important to develop a methodology to select an optimal set of modules that is customized to a business's needs and processes.

2.3. Component EITs and virtual supply chains

Component-based, multi-vendor EIT systems can effectively be linked to the concept of virtual supply chains that are based on newer types of organizations such as virtual corporations, network organizations or value-added partnerships (Sarkis and Sundarraj, in press). Virtual corporations are created by an alliance of independent business processes or enterprises with each contributing its core competency to the network (Byre, 1993; Goldman, 1994; Porter, 1993; Presley et al., 1995; Sheridan, 1993). Similar to virtual corporations, Snow and Miles (1992) introduce the concept of network organizations, where they define dynamic network as a linkage of independent

[3] Although monolithic ERP systems can be (and frequently are) implemented module-by-module, such an implementation would still not be incremental, in the sense that a module has a much larger scope (in terms of costs and impact) than a component does.

companies or partners that perform various functions across the entire supply chain.

A key factor that is emphasized by researchers in designing these supply chain networks is the selection of agile, competent, and compatible partners (Iacocca Institute, 1991; Presley et al., 1995; Snow and Miles, 1992). Davis (1993) describes a global supply chain analysis from raw materials to finished products by highlighting uncertainty at all levels. His work addresses a framework developed at Hewlett-Packard for dealing with uncertainty that adversely affects performance of supplier, manufacturing, and transportation processes. Arntzen et al. (1995) present a global supply chain model (GSCM), which is utilized at Digital Equipment Corporation for designing a production, distribution, and vendor network. Their mixed-integer linear program minimizes cost and/or weighted cumulative production and distribution times subject to a variety of demand and capacity constraints. Talluri et al. (1999) proposed a mathematical model for designing value chain networks by considering efficiencies of potential partners and compatibilities among them. While the efficiencies relate to the performance of processes with respect to their core competencies, the compatibility issues involve integration costs, inception times, and distances among partners.

Since the design of a component-based EIT system involves the procurement and implementation of several software components, each specializing in a particular functional area of business, it can be compared in that sense to the design of a value chain network.

2.4. Literature review of EIT evaluation

A number of models have been proposed for the evaluation of traditional IT acquisitions, by considering them to be a form of a capital purchase. Kelley (1994) discusses the impact of IT on production time per unit, while Hitt and Brynjolfsson (1996) propose productivity, business profitability and consumer surplus as a way of measuring IT value. Farbey et al. (1993) provide a review of various approaches used for IT investment and evaluation. The techniques reviewed by them include basic financial capital budgeting, cost-benefit analysis, information economics, and multi-objective/criteria approaches.

Others have brought out how IT evaluation needs to consider intangible aspects as well. Cronk and Fitzgerald (1999) have discussed how the value of IT systems can be judged from many different perspectives. Torkzadeh and Doll (1999) discuss the impact of IT on task productivity, task innovation, customer satisfaction and management control. Davis (1989), and Bailey and Pearson (1983) consider the impact of IT from the viewpoint of usefulness and currency, respectively. None of the above-mentioned works, however, consider an important unique aspect pertaining to the evaluation of a componentized EIT system. Unlike traditional IT systems that are stand-alone, integration of components is an important requirement to the delivery of an enterprise-wide solution. A model for IT procurement considering integration costs is absent in the literature, and will be focus of this paper.

3. Model formulation

We are given a set of functional areas that are to be integrated with one another. For each area, components are offered by multiple vendors, each with a procurement cost that varies with the different periods of the planning horizon. The cost of integrating two components depends on the functional area, the vendor issuing the component and the time period of integration. The objective is to select a set of components that will minimize the total procurement and integration costs. The notation is now defined for the integer-programming model.

Notation

T	planning horizon $\{1,\ldots,T\}$
I	set of functional areas that need to be covered
J_i	set of components for area i ($i \in I$)
B_t	budget available in period t
p_{jt}	extra discounted revenue (in \$) obtained by implementing component in time t ($t \in T$, $j \in J_i$, $i \in I$)

c_{jt} discounted cost of purchasing component j in time t

$d_{jj't}$ discounted cost of integrating component j with component j' in time t

JJ $\{(j,j')|$ for each $j \in J_i$ $(i \in I)$, $j' \in J_{i'}$ such that $i \neq i'$ and $i' \in I$ $\}$ (i.e., JJ is the set of ordered pairs of components requiring integration)

x_{jt} 1 if component j is chosen to be implemented in period t and 0 otherwise

$y_{jj't}$ 1 if an integration is to be set up between components j and j' in time period t, and 0 otherwise

Model 1

$$\text{Max} \sum_{t\in T, j\in J_i, i\in I} (p_{jt} - c_{jt})x_{jt} - \sum_{t\in T,(j,j')\in \text{JJ}} d_{jj't}y_{jj't} \quad (1)$$

s.t.

$$\sum_{j\in J_i, i\in I} c_{jt}x_{jt} + \sum_{(j,j')\in \text{JJ}} d_{jj't}y_{jj't} \leqslant B_t \quad \text{for all } t \in T, \quad (2)$$

$$\sum_{t\in T, j\in J_i} x_{jt} = 1 \quad \text{for all } i \in I, \quad (3)$$

$$y_{jj't} \geqslant x_{jt} + x_{j'k} - 1 \quad \text{for all } (j,j') \in \text{JJ}, \quad t \in T,\ k \leqslant t, \quad (4)$$

$$y_{jj't} \geqslant x_{jk} + x_{j't} - 1 \quad \text{for all } (j,j') \in \text{JJ}, \quad t \in T,\ k < t, \quad (5)$$

$$y_{jj't}, x_{jt} \in \{0,1\} \quad \text{for all } t \in T,\ j,j' \in J_i,\ i \in I. \quad (6)$$

This model maximizes the net present profit (extra net revenue − purchase cost − integration costs, discounted to the current period). Constraint (2) specifies the budgetary requirement, and constraint (3) specifies that every functional area must be covered. Constraints (4) and (5) ensure that integration costs are incurred when two components that need to be integrated (i.e., belonging to set JJ) are selected. Also, since the objective maximizes negative $y_{jj'}$, it is also implicitly ensured that integration costs are incurred only when the corresponding systems are selected.

4. Experiments with model

We now report managerial implications derived from experimenting with over 100 problem instances of our model. Three experiments were conducted:

- changing integration costs
- varying the number of time periods, keeping budget per quarter constant
- varying the number of time periods and fixing the overall budget

In Section 4.1, we describe the basic structure of the problem that was used in our experiments, while Section 4.2 discusses the experiments along with their implications.

4.1. Problem data

Our problem instances involved three functional areas, with three alternatives for the first area, two for the second and four for the third, denoted by $J_1 = \{\text{A}, \text{B}, \text{C}\}$, $J_2 = \{\text{D}, \text{E}\}$ and $J_3 = \{\text{F}, \text{G}, \text{H}, \text{I}\}$, respectively. Assuming that all the functional areas need to be integrated with one another, there will be 26 possible integration costs for each time period (e.g., A with D through I, B with D through I, etc., Table 3 depicts all these options). These integration costs are drawn from the Uniform distribution $U(1,7)$, while purchase costs and revenues were drawn from $U(8,10)$ and $U(15,25)$, respectively (the notation $U(a,b)$ is used to represent uniform random numbers in the range a and b). Tables 1–3 depict the generated data for purchase costs, revenues, and integration costs, respectively.

We now report on the experiments with this data set.

4.2. Experiments

4.2.1. Changing integration costs

The components that are purchased must be integrated with one another in order to provide an enterprise solution. Integration costs reflect the extent to which a component can be standardized. More standardization implies lower integration

Table 1
System procurement costs by quarter

System	Quarter									
	1	2	3	4	5	6	7	8	9	10
A	12	14	13	10	10	14	14	13	12	10
B	9	12	10	11	9	14	10	12	8	7
C	8	11	9	14	11	13	12	12	10	6
D	9	11	13	14	11	14	13	9	12	8
E	12	8	11	11	13	10	13	11	6	12
F	11	12	14	10	11	13	9	12	7	14
G	8	9	11	8	8	9	9	13	11	10
H	12	9	9	8	10	8	12	12	9	8
I	9	12	10	12	13	10	14	11	17	11

Table 2
System revenues by quarter

System	Quarter									
	1	2	3	4	5	6	7	8	9	10
A	23	16	16	19	22	20	25	22	24	25
B	15	22	18	17	16	19	20	16	18	17
C	24	19	23	23	24	19	24	15	15	18
D	21	20	19	23	19	25	19	17	20	20
E	21	21	25	16	23	25	24	16	15	18
F	16	21	19	23	17	18	23	16	21	22
G	25	24	19	23	17	16	23	23	15	18
H	19	23	18	20	23	21	23	17	16	17
I	21	25	20	24	20	23	16	21	23	21

Table 3
System integration costs by quarter

System integration	Quarter									
	1	2	3	4	5	6	7	8	9	10
AD	7	4	6	7	1	7	5	3	1	2
AE	6	6	2	2	1	6	5	5	3	2
AF	3	1	3	2	2	2	6	5	2	2
AG	6	2	5	4	2	3	2	7	4	3
AH	5	3	5	5	5	1	7	1	6	1
AI	5	3	3	7	1	6	3	3	3	1
BD	2	4	3	5	5	6	3	6	4	1
BE	7	7	5	1	1	7	4	5	1	3
BF	2	5	2	4	1	4	5	2	1	2
BG	5	2	3	5	4	7	1	2	1	1
BH	4	2	2	1	2	1	1	2	6	4
BI	2	6	7	1	7	6	3	7	5	5
CD	3	3	5	6	1	3	3	7	3	6
CE	2	4	4	6	3	7	3	1	2	4
CF	4	1	5	6	5	2	3	6	1	3
CG	3	6	1	1	5	4	7	4	1	2
CH	1	3	2	7	4	5	7	7	4	1
CI	6	2	3	3	3	1	6	2	2	6
DF	6	2	3	7	7	1	5	5	2	7
DG	1	7	5	3	7	7	4	3	2	4
DH	1	7	1	6	6	1	5	6	1	2
DI	1	7	2	6	5	3	4	7	3	1
EF	7	2	7	6	2	7	1	4	4	1
EG	5	4	4	4	7	4	7	7	2	3
EH	6	2	5	7	4	4	5	4	1	2
EI	2	6	4	2	2	2	7	7	2	1

costs, but also lesser scope of differentiation among components from multiple vendors. While inter-component integration costs are substantially lower than what can be expected with monolithic EITs, they are certainly large enough to be considered in the decision-making process. Research shows that integration expenses may represent at least 25% of the system costs (Altman and Pond, 1999; Jilovec, 1999), although research in software engineering could substantially lower these costs in the future.

Our first experiment varies integration costs by setting them at 75%, 50%, 25% and 0% of the values found in Table 3. In order to illustrate the effects of these changes, we considered an 8-period planning horizon with a budget level of 13 per quarter. Clearly, as shown in Table 4, profits can be expected to rise with decreases in integration costs. A closer look at the solutions revealed that in every case, the increase in profits was always slightly greater than the decrease in integration costs. Fig. 2, which plots the total integration costs for each instance along with the corresponding total profits, depicts this result.

From a system procurement standpoint the decrease in integration costs resulted in selecting systems C, E, and G for all four cases (75–0%). However, it is interesting to note the differences between the quarters in which these systems are selected across the four cases. For example, with

Table 4
Results from changing integration costs

Integration costs (%)	Solution	Profit ($)
100	C1, E2, F7	35
75	C3, E2, G1	37.25
50	C1, E3, G4	40.50
25	C3, E6, G1	43
0	C1, E6, G2	46

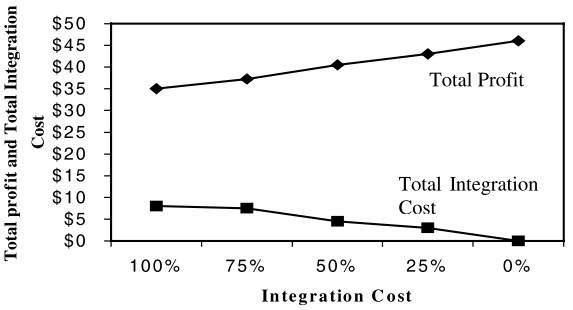

Fig. 2. Effect of varying integration costs.

75% integration costs, system C is selected in quarter 3, system E is selected in quarter 2, and system G is selected in quarter 1 (represented as C3, E2, and G1), which is different from the results of 0% integration costs that identified C1, E6, and G2 as the optimal solution. It is also worth noting that system F, which is selected for 100% integration cost, is dropped from selection in all the other four cases. This type of sensitivity analysis assists decision-makers in effectively planning for the procurement of EIT systems in the presence of changing integration costs.

4.2.2. Varying planning horizon and constant budget per quarter

In a number of situations, returns on various financial investments might determine the cash flow into an organization, and in turn the quarterly budget amount that can be allocated to the EIT project. Project managers must plan in keeping with this reality, and by considering that EIT implementations are quite often prone to delays. Hence, the situation of constant cash flow per quarter and varying time periods becomes relevant.

We vary the planning horizon from 8 to 10 periods, while keeping the budget per quarter at a certain level. This analysis was done for various levels of quarterly budget, ranging from 10 to 32. The results of the analysis are shown in Table 5. It is interesting to note that, for budget levels of 14

Table 5
Results from constant quarterly budget and variable planning horizon

Budget/ Quarter ($)	8-period case		9-period case		10-period case	
	Solution	Profit	Solution	Profit	Solution	Profit
<10	Infeasible	N/A	Infeasible	N/A	Infeasible	N/A
10	Infeasible	N/A	B9, E6, F7	36	B9, E6, F7	36
11	A5, D1, H6	34	B9, E6, F7	36	B9, E6, F7	36
12	A5, D1, H6	34	B9, E6, F7	36	B9, E6, F7	36
13	C1, E2, F7	35	B9, E6, F7	36	A10, E6, I2	38
14	C1, E2, G4	35	B9, E6, F7	36	A10, E6, F7	39
15	C1, E3, F7	36	B9, E6, F7	36	A10, E6, F7	39
16	C1, E3, F7	36	B9, E6, F7	36	A10, E6, F7	39
17	A5, E3, G1	36	B9, E6, F7	36	A10, E6, F7	39
18	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
19	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
20	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
21	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
22	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
23	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
24	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
25	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
26	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
27	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
28	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
29	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
30	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
31	A5, D1, G1	37	A5, D1, G1	37	A10, E6, F7	39
32	C1, D1, G1	38	C1, D1, G1	38	A10, E6, F7	39
>32	C1, D1, G1	38	C1, D1, G1	38	A10, E6, F7	39

and above, the 10-period implementation strategy for this illustrative application has resulted in higher profit levels when compared to 8- and 9-period implementation time frame. For budget levels of 11–13, the results for the 9- and 10-period cases are indifferent from each other, but better than the 8-period case. Finally, for a budget level of 10, the 8-period case is infeasible, while the 9- and 10-period scenarios yield the same results.

Some managerial decisions that can be made from this analysis include the identification of minimum and maximum levels of budget required per quarter, and the selection of an optimal implementation time frame for a given budget level.

4.2.3. Varying planning horizon and fixed overall budget

This experiment relates to situations in which an overall budget for the implementation is fixed a priori and an optimal planning horizon for the implementation is desired. This is very important for a number of medium- and small-sized companies that face severe limitations on the amount that can be spent on an EIT implementation.

We changed the planning horizon from 8 to 10 quarters and the overall budget from 88 to 288 in steps of 8. For each budget value, the allocation per quarter was equally divided among the number of quarters, as shown in Table 6. The budget per quarter and the profit (= revenue − purchase cost − integration cost) for each of the three time periods are shown in Table 6. For overall budget levels of 136 or higher, the 10-period implementation strategy is the optimal with a profit value of 39. Also, it is interesting to note that at lower levels of overall budget the 9- and 10-period cases yield the same results. Finally, at the lowest possible level of overall budget value of 88 only the 8-period implementation is feasible. These types of insights assist companies with severe budget constraints in identifying an optimal procurement plan and implementation strategy for EIT systems.

5. On allocating a project budget

In this section, we discuss an extension that considers a practical issue faced by decision-makers. Since an EIT implementation frequently spans multiple years and is led by a point-person with visibility throughout the organization, the question of how to allocate the project budget becomes important. One method involves treating each quarter separately and allocating a budget to each, thereby providing more fiscal control over the project. Alternatively, the project manager could be allocated an overall *project budget*, with the flexibility to expend the monies as and when the need arises. The second scenario allows budget surpluses in one period to be used in the next, but does not give financial managers a tighter control over the project. To test how the second scenario would work, we consider the following model, where s_t represents the budget leftover from period t (we set $s_0 = 0$).

Model 2

$$\text{Max} \sum_{t\in T, j\in J_i, i\in I} (p_{jt} - c_{jt})x_{jt} - \sum_{t\in T,(j,j')\in \mathbf{JJ}} d_{jj't}y_{jj't}$$

$$\text{s.t.} \sum_{j\in J_i, i\in I} c_{jt}x_{jt} + \sum_{(j,j')\in \mathbf{JJ}} d_{jj't}y_{jj't} + s_t - s_{t-1} = B_t \quad \text{for all } t \in T, \tag{7}$$

$$\sum_{t\in T, j\in J_i} x_{jt} = 1 \quad \text{for all } i \in I,$$

$$y_{jj't} \geqslant x_{jt} + x_{j'k} - 1 \quad \text{for all } (j,j') \in \mathbf{JJ},\ t \in T,\ k \leqslant t,$$

$$y_{jj't} \geqslant x_{jk} + x_{j't} - 1 \quad \text{for all } (j,j') \in \mathbf{JJ},\ t \in T,\ k < t,$$

$$y_{jj't}, x_{jt} \in \{0,1\} \quad \text{for all } t \in T,\ j,j' \in J_i,\ i \in I.$$

We evaluated the above model for the case of varying planning horizon and fixed budget and the results are shown in Table 7. The bold numbers in the table indicate the cases in which the profit increases, as compared to Table 6. This implies that the allocation of a project budget must be considered when the resources are scarce.

Table 6
Results from fixed overall budget and varying planning horizon

8-period case		9-period case		10-period case	
Budget/ Quarter	Profit	Budget/ Quarter	Profit	Budget/ Quarter	Profit
11	34	9.78	N/A	8.80	N/A
12	34	10.67	36	9.60	N/A
13	35	11.56	36	10.40	36
14	35	12.44	36	11.20	36
15	36	13.33	36	12.00	36
16	36	14.22	36	12.80	36
17	36	15.11	36	13.60	38
18	37	16.00	36	14.40	39
19	37	16.89	36	15.20	39
20	37	17.78	36	16.00	39
21	37	18.67	37	16.80	39
22	37	19.56	37	17.60	39
23	37	20.44	37	18.40	39
24	37	21.33	37	19.20	39
25	37	22.22	37	20.00	39
26	37	23.11	37	20.80	39
27	37	24.00	37	21.60	39
28	37	24.89	37	22.40	39
29	37	25.78	37	23.20	39
30	37	26.67	37	24.00	39
31	37	27.56	37	24.80	39
32	38	28.44	37	25.60	39
33	38	29.33	37	26.40	39
34	38	30.22	37	27.20	39
35	38	31.11	37	28.00	39
36	38	32.00	38	28.80	39
>36	38	>32	38	>28.8	39

Table 7
Results from fixed overall budget and varying planning horizon with budget slacks

8-period case		9-period case		10-period case	
Budget/ Quarter	Profit	Budget/ Quarter	Profit	Budget/ Quarter	Profit
11	**36**	9.78	**36**	8.80	**39**
12	**36**	10.67	36	9.60	**39**
13	**36**	11.56	36	10.40	**39**
14	**36**	12.44	36	11.20	**39**
15	36	13.33	36	12.00	**39**
16	36	14.22	36	12.80	**39**
17	36	15.11	36	13.60	**39**
18	37	16.00	36	14.40	39
19	37	16.89	36	15.20	39
20	37	17.78	36	16.00	39
21	37	18.67	37	16.80	39
22	37	19.56	37	17.60	39
23	37	20.44	37	18.40	39
24	37	21.33	37	19.20	39
25	37	22.22	37	20.00	39
26	37	23.11	37	20.80	39
27	37	24.00	37	21.60	39
28	37	24.89	37	22.40	39
29	37	25.78	37	23.20	39
30	37	26.67	37	24.00	39
31	37	27.56	37	24.80	39
32	38	28.44	37	25.60	39
33	38	29.33	37	26.40	39
34	38	30.22	37	27.20	39
35	38	31.11	37	28.00	39
36	38	32.00	38	28.80	39
>36	38	>32	38	>28.8	39

6. Conclusions and extensions

Component-based systems are emerging information technologies that can help resolve the dilemma of choosing between a protracted implementation of a traditional, single-vendor enterprise system and the problems of working with an uncoordinated supply chain. Such systems offer organizations a number of advantages, including the ability to choose components that best match the company's business processes, to tailor components to the individual needs of business units within the organization, and to incrementally pay for the overall system. We have presented a multi-period optimization model to determine a set of components that can provide an enterprise solution. We consider procurement costs as well as inter-component integration costs that arise from the need to provide information across the supply chain. Experiments have been conducted to show the utility of the model for making decisions concerning budget levels and the time period of implementation. Such an optimization model for an important practical problem has not been proposed in the literature.

A number of extensions are possible with the work described herein. Clearly, the model requires the collection of various types of data, followed by cost and revenue forecasts that are likely subject to errors. Hence, managerially, it is important to understand the sensitivity of the results to changes in problem parameters. A second extension could be to formulate other decisions mathematically. The current model assumes that integration

between two components takes place as soon as both are purchased. It would be interesting to see if this is indeed optimal, by incorporating decision variables that indicate the timing of the integration task. A third extension can involve the incorporation of qualitative factors such as availability of the right work force, skill set in the work force, their motivation level, competition, level of program maturity in an organization, which all facilitate the integration process. Finally, since the model is large in size, efficient solution algorithms would need to be developed.

References

Alderson, A., Shah, H., 1999. Viewpoints of legacy systems. Communications of the ACM 42 (3), 115.

Appleton, E., 1997. How to survive ERP. Datamation 43 (3), 50–53.

Altman, R., Pond, K., 1999. Components in the ERP World. Research Note, Gartner Advisory. http://www.isr.bucknell.edu/gartner/research/ras/78600/78643/78643.html.

Arntzen, B.C., Brown, G.G., Harrision, T.P., Trafton, L.L., 1995. Global supply chain management at Digital Equipment Corporation. Interfaces 25 (1), 69–93.

Anonymous, 1999. ERP averages. Computerworld, April 5, p. 6.

Bailey, J., Pearson, S., 1983. Development of a tool for measuring and analyzing user satisfaction. Management Science 29 (5), 530–545.

Baumer, D. et al., 1997. Framework development for large systems. Communications of the ACM 40 (10), 52–59.

Beugnard, A., Jezequel, J., Plozeau, N., Watkins, D., 1999. Making components contract aware. IEEE Computer, 38–44.

Boehm, B., 1999. Managing software productivity and reuse. Computer, 111–113.

Brown, A., Wallnau, K., 1996. Engineering of component-based systems. In: Brown, A.W. (Ed.), Component-based Software Engineering: Selected Papers from the Software Engineering Institute. IEEE Computer Society Press, Los Alamitos, CA.

Brown, A., Wallnau, K., 1998. The current state of CBSE. IEEE Software, 37–46.

Brynjolfsson, E., 1993. The productivity paradox of information technology. Communications of the ACM 36 (12), 67–77.

Burd, E., Munro, M., 1998. A method for the identification of reusable units through the reengineering of legacy code. Journal of Systems and Software 44 (2), 121.

Byre, A.J., 1993. The virtual corporation. Business Week, 98–103.

Chopra, S., Meindl, P., 2001. Supply Chain Management: Strategy, Planning, and Operation, first ed. Prentice Hall, New Jersey.

Cliff, S., 1999. ERP implementation. Harvard Business Review 77 (1), 16.

Clements, P., 1996. From subroutines to subsystems: Component-based software development. In: Brown, A.W. (Ed.), Component-based Software Engineering: Selected Papers from the Software Engineering Institute. IEEE Computer Society Press, Los Alamitos, CA.

Cronk, M., Fitzgerald, E., 1999. Understanding 'IS business value': Derivation of dimensions. Logistics Information Management 12 (1–2), 40–49.

Davenport, T., 1998. Putting the enterprise into enterprise systems. Harvard Business Review, 121–131.

Davis, F., 1989. Perceived usefulness, perceived ease of use and user acceptance of information technology. MIS Quarterly 13 (3), 319–342.

Davis, T., 1993. Effective supply chain management. Sloan Management Review 34 (4), 35–46.

Demeyer, S., Meijler, T., Nierstrasz, O., Steyaert, P., 1997. Design guidelines for tailorable frameworks. Communications of the ACM 40 (10), 60–64.

Fan, M., Stallaert, J., Whinston, A.B., 2000. The adoption and design methodologies of component-based enterprise systems. European Journal of Information Systems 9, 25–35.

Farbey, B., Land, F., Targett, D., 1993. IT Investments: A study of methods and practices. Butterworth-Heinemann, Oxford.

Fingar, P., 2000. Component-based frameworks for e-commerce. Communications of the ACM 43 (10), 61–66.

Garlan, D., Allen, R., Ockerbloom, J., 1995. Architectural mismatch: Why reuse is hard, November.

Goldman, L.S., 1994. Co-operating to compete. CMA Magazine 68 (2), 13–17.

Hitt, L., Brynjolfsson, E., 1996. Productivity, business profitability and consumer surplus: Three different measures of information technology value. MIS Quarterly 20 (2), 121–142.

Holt, S., 1999. Competition heats up in ERP. InfoWorld 21 (6), 65.

Iacocca Institute, 1991. 21st Century manufacturing strategy, Lehigh University, Bethlehem, PA.

Jeong, H. et al., 2001. Design of a software component bank for distribution. Journal of Systems Integration 10, 223–237.

Jilovec, N., 1999. Preparing for the electronic marketplace. ERP World Conference Proceedings. www.erpworld.org/conference/erpe99/proceedings/021.

Kelley, M.R., 1994. Productivity and information technology: The elusive connection. Management Science 40, 1406–1425.

Kremers, M., Van Dissel, H., 2000. ERP systems migrations. Communications of the ACM 43 (4), 52–57.

Kumar, K., Van Hillegersberg, J., 2000. ERP experiences and evolution. Communications of the ACM 43 (4), 22–26.

Lunney, T., McCaughey, A., 2000. Component based distributed systems—CORBA and EJB in context. Computers and Physics Communication 127, 207–214.

Markosian, L. et al., 1994. Using an enabling technology to reengineer legacy systems. Communications of the ACM 37 (5), 58–70.

Orfali, R., Harkey, D., Edwards, J., 1996. The essential distributed objects survival guide. John Wiley, London.

Presley, A., Barnett, B., Liles, D.H., 1995. A virtual enterprise architecture. In: Fourth Annual Agility Forum Conference, vol. 2, p. 312.

Porter, A.L., 1993. Virtual companies reconsidered. Technology Analysis and Strategic Management 5 (4), 413–420.

Sarkis, J., Sundarraj, R.P., in press. Evolution of brokering paradigms in E-commerce enabled manufacturing, International Journal of Production Economics.

Sawyer, S., 2000. Packaged software: Implications of the differences from custom approaches to software development. European Journal of Information Systems 9, 47–58.

Sheridan, J.H., 1993. Agile manufacturing: A new paradigm. International Productivity Journal 11, 39–48.

Snow, C.C., Miles, R.E., 1992. Managing 21st century network organizations. Organizational Dynamics 20 (3), 5–20.

Soh, C., Kien, S., Tay-Yap, J., 2000. Cultural fits and misfits: Is ERP a universal solution? Communications of the ACM 43 (4), 47–53.

Sousa, J.P., Garlan, D., 2001. Formal modeling of enterprise JavaBeans component integration framework. Information and Software Technology 43, 171–188.

Sprott, D., 2000. Componentizing the enterprise application packages. Communications of the ACM 43 (4), 63–69.

Staringer, W., 1994. Constructing applications from reusable components. IEEE Software, 61–68.

Stets, R., Hunt, G., Scott, M., 1999. Component-based APIs for versioning distributed applications. IEEE Computer, 54–61.

Talluri, S., Baker, R.C., Sarkis, J., 1999. A framework for designing efficient value chain networks. International Journal of Production Economics 62, 133–144.

Torkzadeh, G., Doll, W., 1999. The development of a tool for measuring the perceived impact of information technology on work. OMEGA: The International Journal of Management Science 27, 327–339.

Trunick, P.A., 1999. ERP: Promises or pipe dreams. Transportation and Distribution 40 (1), 23.

Vitharna, P., Jain, H., 2000. Research issues in testing business components. Information and Management 37, 297–309.