Chapter 10

The Enterprise System Experience—From Adoption to Success

M. Lynne Markus and Cornelis Tanis

"[T]he notion that a company can and ought to have an expert (or a group of experts) create for it a single, completely integrated supersystem—an 'MIS'—to help it govern every aspect of its activity is absurd." (Dearden, 1972, p. 101)

For some 20 years after John Dearden wrote these words, history proved him right. Today, however, there is a booming market for software packages claiming to provide a total, integrated solution to companies' information-processing needs. Even companies that choose not to adopt such packages are pursuing aggressive strategies of systems integration by redeveloping custom software and adopting technologies such as data warehousing. Integrated enterprise systems deserve serious research attention because of their great potential for financial, technical, managerial, human, and strategic benefits, costs, and risks.

This chapter provides a theoretical framework for analyzing, both retrospectively and prospectively, the business value of enterprise systems. We first describe the historical context in which enterprise systems emerged. Next we identify the key characteristics of enterprise systems, discuss the reasons companies do and do not adopt them, and summarize arguments about why enterprise systems are an important topic for research. We then analyze enterprise systems in terms of the concept of success. We argue that the many facets of success create difficulties for

Acknowledgments: The authors gratefully acknowledge the support of other members of their research team: Sheryl Axline, Lars Bo Eriksen, and Dave Petrie. In addition, they wish to thank the following individuals for helpful input at various stages in this work: Carole Agres, Matt Alvarez, David Brown, Dorothy Cooney, Sabine Hirt, Debbie Mahan, Christina Soh, and Nancy Wendt.

both academics and practitioners and require a framework for understanding the "enterprise system experience." Essential elements of this framework include phases, starting conditions, goals, plans, and quality of execution. We conclude with suggestions for future research.

HISTORICAL CONTEXT OF ENTERPRISE SYSTEMS

Whether because the capacity of computers and programming languages was too small or organizations were content to manage themselves along narrow functional lines, the 1970s vision of a single integrated information system for the enterprise remained a mirage for the majority of computer-using organizations. Instead, organizations created "islands of automation" (McKenney & McFarlan, 1982). When companies identified a new application for IT, they programmed a discrete new information system. If the new system had something in common with existing systems, it was loosely interfaced (sometimes manually) rather than integrated tightly with these existing systems. As a result, combining information about sales or manufacturing with accounting data was difficult and error prone. Analyses could be performed easily only at a summary level; detailed transaction-level analysis required special ad hoc programming or manual record sifting. Multiple systems contained the same data elements; because duplicate data entry promoted errors and because systems varied in update schedules, the data in different systems often did not agree. Decision making was stymied. Corporate restructurings necessitated major reprogramming (or were abandoned as technologically infeasible[1]). The total organizational costs of maintaining this loose patchwork of redundant and overlapping systems grew, eclipsing the funds available for building new ones (Lientz & Swanson, 1980). In short, organizations finally began to experience the burden of *not* having pursued the dream of "one-company, one-system."

But the dream did not die among members of the information systems community. Throughout the 1980s and 1990s, software entrepreneurs in Germany, the Netherlands, the United States, and elsewhere were developing integrated software packages in which multiple functional applications shared a common database.[2] A single transaction such as a sales order could "flow through" the entire applications suite, automatically updating financial and inventory records without additional data entry and feeding various planning and decision support systems. As the vendors enhanced the functionality of their integrated packages, they claimed with some (but not total) truth that their products met all the information-processing needs of the companies that adopted them. These packages became known as enterprise resource planning (ERP) systems, and they include some that have developed out of the administrative (financial and human resources) side of the business (e.g., SAP and Peoplesoft), as well as some that grew from materials resource planning in manufacturing (e.g., Baan).

Software packages have always been more acceptable for smaller enterprises than for large ones for a variety of historical, technical, and economic reasons. Therefore, integrated packages made relatively little headway in the largest organizations until the mid-1990s, when vendors began to offer versions for the client/server architecture. For some time, companies had been aware that the client/server architecture afforded advantages relative to mainframe-based applications. The latter had much higher operating costs, and they did not support graphical user interfaces for business

applications. Many companies tried redeveloping their core transaction systems for the client/server architecture, only to discover that the process was extremely expensive and highly failure prone; packages as an alternative to in-house (re)development started to become an appealing option, even in large companies. Packages got a huge boost as companies began to realize the full impact of the Year 2000 (Y2K) problem—and came to see packages as a solution. (Other reasons for ERP adoption are discussed later.) Suddenly, nearly everyone got on the ERP bandwagon.

By 1998 approximately 40% of companies with annual revenues of more than $1 billion had implemented ERP systems (Caldwell & Stein, 1998). ERP vendors began aggressively targeting smaller firms because they represent a much larger market. In aggregate, the ERP marketplace is huge. SAP Inc., the largest ERP system vendor, had 1997 revenues of $3.3 billion (Davenport, 1998). A research firm recently predicted that the ERP market would reach $66.6 billion by 2003 (AMR Research, 1999).[3] The ERP systems and services market has cooled somewhat at present because many companies have already implemented ERP in response to Y2K concerns. Nevertheless, interest in related enterprise systems such as sales force automation, supply chain integration, and product configuration remains strong.

Enterprise systems are clearly a phenomenon in the IT marketplace. Their potential significance for computer-using organizations cannot be overstated. They represent a nearly complete rearchitecting of an organization's portfolio of transactions-processing applications systems to achieve integration of business processes, systems, and information—along with corresponding changes in the supporting computing platform (hardware, software, databases, telecommunications). On a societal scale, the enterprise system phenomenon seems to be about a complete renewal of enterprise IT infrastructure.[4] The potential consequences—economic, technical, and social—of this development are indeed significant. Two small examples will illustrate this point. One ERP implementation enabled a 70% reduction in accounting personnel by eliminating duplicate data entry and many consolidation tasks (Larsen & Myers, 1997). And analysts have speculated that widespread adoption of the same ERP package by the firms in a single industry (an observed phenomenon for semiconductor manufacturers) might lead to the elimination of process innovation–based competitive advantage (Davenport, 1998).

Despite the potential benefits of widespread IT renewal, the process to date has been far from smooth. Some organizations have utterly failed in their attempts to install ERP systems (Bulkeley, 1996). Others have achieved some benefits despite decidedly rocky beginnings (Cole-Gomolski, 1998; Stedman, 1998a, 1998b, and 1998c; Stedman, 1999b). Many have failed to achieve the hoped-for financial returns on their ERP investment (KPMG, 1998; Stedman, 1999a).

Of course, companies have had similar difficulties with each new wave of information technology since the first mainframe systems (McKenney, Copeland, & Mason, 1995). It takes years at best to realize some envisioned IT-enabled changes in organizational processes and performance, and there are many ways to fail along the way.[5] So one wants to know how, if at all, the enterprise system *experience* is similar to, or different from, experiences with any other type of information technology, such as decision support groupware, and workflow. In the IS field, theory and empirical findings related to this question are highly dispersed. Similar concepts are discussed under different names, such as implementation and adoption. Conceptually related ideas have generated literatures with little overlap; for example, the research literature on IT impacts remains quite distinct from the literatures on system development methodologies and on the payoffs from IT investments. Because the enterprise system

experience cuts across all of these "islands of ideation," it is time to take an integrated view. This chapter attempts an integration of the diverse bodies of knowledge bearing on the enterprise system phenomenon.

ENTERPRISE SYSTEMS

In this section, we identify the key characteristics of enterprise systems, list reasons companies do and do not adopt them, and summarize arguments about why enterprise systems are an important topic for research. Enterprise systems are commercial software packages that enable the integration of transactions-oriented data and business processes throughout an organization (and perhaps eventually throughout the entire interorganizational supply chain). In our definition, enterprise systems include ERP software and such related packages as advanced planning and scheduling, sales force automation, customer relationship management, and product configuration. Organizations that adopt enterprise systems have a wide range of options for implementation and ongoing operations, from do it yourself, through selective external assistance, to total outsourcing.

CHARACTERISTICS OF ENTERPRISE SYSTEMS

Enterprise systems have several characteristics, each with important implications for the organizations that adopt them.

- *Integration.* Enterprise systems promise "seamless integration of all the information flowing through a company—financial and accounting information, human resource information, supply chain information, and customer information" (Davenport, 1998, p. 121). However, it is extremely important to note that achieving this integration depends on "configuring" (setting up) the system in particular ways. Configuration in this context[6] means choosing which package modules to install and setting software parameters to represent, for example, the company's products, customers, accounts and the particular arrangement of business processes, such as centralized or decentralized warehousing and purchasing. An additional important part of the configuration task is capturing configuration decisions and their rationale. The quality of such documentation is essential to the organization's ability to make future changes efficiently and effectively.
    - It is possible, especially in large, complex organizations, to configure enterprise systems so that the benefits of integration are not achieved. For example, companies may purchase and install only the "financials" modules of an enterprise system, thus depriving themselves of the potential advantages of integrating accounting data with sales, manufacturing, and distribution data. Furthermore, an organization may allow each of its business units to adopt a different enterprise system or to configure the same enterprise system however they see fit, with the result that it is not possible to obtain integration benefits from common purchasing or better decision making.[7]
- *Packages.* Enterprise systems are commercial packages; that is, they are purchased or leased from software vendors rather than being developed in-house

from scratch. This has two important implications for the organizations that adopt them.

- First, the IS life cycle is different.[8] Rather than designing a system to meet the organization's idiosyncratic ways of working, the adopters of an enterprise system often adjust the organization's ways of working to fit the package (because modifying packages has numerous negative consequences). Consequently, package adopters sometimes forgo or curtail the analysis of current information requirements and business processes that is a hallmark of the traditional IS life cycle. Furthermore, the process of configuring an enterprise system for an organization differs substantially from software programming. Programming involves creating new software functionality. Configuration involves adapting the generic functionality of a package to the needs of a particular organization (usually by setting parameters in tables). Configuration is often performed by teams of end-users with IS specialists working primarily on infrastructural issues.[9] In other words, enterprise package implementation obsoletes some of the IT skills commonly found in IT adopting organizations and requires the acquisition of new skills. In particular, enterprise systems put a premium on skills related to (1) mapping organizational requirements to the processes and terminology employed by the vendor and (2) making informed choices about the parameter setting.[10]
- Second, organizations that purchase an enterprise system enter into long-term relationships with software vendors. It is true that some organizations purchase an enterprise system with the idea that they will modify the packages to suit idiosyncratic needs. But doing so reduces their ability to benefit from vendors' continued development of the packages, and it may create dependency on outside contractors who specialize in enterprise software customizations. (Vendors generally do not undertake to support or maintain customers' modifications of their software.) Consequently, many organizations depend on the vendor for continued enhancement of the package (for example, redeveloping the software for future computing architectures). As a result, purchasers of an enterprise system may need to become active in user organizations, a mechanism by which software buyers collectively try to influence the vendor's plans for package maintenance and enhancement. Because of their dependence on vendors, organizations are vulnerable in the event that their chosen vendor goes out of business or lacks the resources for continued technical development. Furthermore, they are committing themselves to upgrading the software periodically if they hope to avoid major conversion headaches.[11]

- *Best Practices.* Because they are designed to fit the needs of many organizations, enterprise systems are built to support generic business processes that may differ quite substantially from the way any particular organization does business. By talking to many businesses and looking to academic theory (or to APICS)[12] about the best way to do accounting or manage a production floor, the vendors of enterprise systems have crafted what they claim to be "best practices."[13] Best practices represent a powerful reason to adopt enterprise systems without modifying them because few organizations claim to have redesigned all their business processes for cross-functional efficiency and effectiveness—which was the stated purpose of business process reengineering (Hammer, 1990). But to realize the advantages of the

best practices embedded in enterprise systems, most adopting organizations must commit themselves to some degree of business process reengineering (Connolly, 1999).[14]

- There is great debate about the relative advantage of doing business process reengineering[15] before, during, or after enterprise system implementation. But there is general consensus that business process change adds considerably to the expense and risk of the implementation of enterprise systems. The principal reason is the difficulty of managing large-scale human and organizational change. For example, four business functions are involved in the early phases of aircraft design and manufacture: contract management, project accounting, project management, and estimating. In traditionally organized aerospace firms, these four functions may be performed by different organizational units with consequent recycling and layers of approval. Baan's software for aerospace and defense industries *requires* (by means of screen design and software processing logic) that all four functions be performed by an "integrated product team" comprising all four sets of skills. Enterprise systems have many such embedded business practices, the details of which may vary from vendor to vendor. Some organizations rebel against the inflexibility of these imposed business practices; even when organizational leadership accepts the need for change, the process of implementing enterprise systems can involve considerable change in organizational structure, job design, work sequencing, training, and so on.

- *Some Assembly Required*. At one level, the claim of enterprise systems to be "integrated" is wildly overstated. What is integrated is the *software*, not the computing *platform* on which it runs. Empirically, enterprise system–adopting companies have had great difficulty integrating their enterprise software with a package of hardware, operating systems, database management systems software, and telecommunications suited to their particular organizational size, structure, and geographic distribution.[16] And this is only one of the integration challenges associated with enterprise systems. Marketing claims aside, in today's state of the art, no single enterprise system meets all the information-processing needs of the majority of organizations.

  - In many cases, enterprise system–adopting organizations will need to interface the package to the company's own proprietary "legacy" systems, for which the enterprise system does not provide an adequate replacement. The organizations may also need to acquire and interface the package to any number of "bolt-on" applications from third-party vendors for various tasks.[17] Sometimes the adopting organization may turn to a third party that has integrated the enterprise package around the special needs of a particular industry segment.[18] Finally, some organizations adopt a "best-of-breed" strategy in which they try to integrate several enterprise packages from different vendors, each designed to be the best fit in its class with the needs of the adopting organizations. Examples of companies that have adopted the best-of-breed approach are American Standard Companies (Bashein et al., 1997) and Starbucks (Aragon, 1997).

- *Evolving*. Finally, like all of IT, enterprise systems are rapidly changing. First, they are changing *architecturally*. In the 1980s enterprise systems were designed for the mainframe system architecture. Today, they are designed for

client-server architectures. Some vendors have just released Web-enabled versions of the software; most vendors have object-oriented versions under development. Baan is pursuing a strategy of "componentization," consisting of an open backbone to which the offerings of other vendors can be connected. The *functionality* of enterprise systems is also evolving. Today, enterprise software vendors are releasing extensions to their core products[19] designed to handle "front office" (i.e., sales management), "supply chain" (i.e., advanced planning and scheduling), data warehousing, specialized vertical industry solutions, and other functions. Enhancements such as customer relationship management and electronic commerce are in the works. *Service arrangements* are also changing. Some services firms offer packaged implementation services; others (often called application service providers) are offering ongoing enterprise software functionality on an outsourced basis. Enterprise systems *terminology* will undoubtedly change, too; only time will tell whether the extensions continue to be identified as something different from enterprise systems or are eventually folded into the enterprise system rubric.

- However things are called, many people now regard enterprise systems (or enterprise integration achieved in other ways) as *the* organizational infrastructure that will support future value-generating applications, such as linking the organization's operations with those of suppliers and customers, leading to substantial reductions in duplicated activities *across* firms.

REASONS FOR ADOPTING ENTERPRISE SYSTEMS

Given the richness of enterprise systems in terms of functionality and potential benefits to adopting organizations, it should not be surprising that companies are adopting these systems for many different reasons[20] (Ross, 1999). Some companies have largely technical reasons for investing in enterprise systems. Examples are the desire to reduce mainframe system operating costs, the need to solve the Y2K and similar problems, the need for increased systems capacity to handle growth, or the need to solve the maintenance headaches associated with aging legacy systems. Other companies give largely business reasons for adopting enterprise systems. For example, the company may need but not have, due to limitations in its legacy systems, the ability to present "one face to the customer" or to know whether it has finished goods inventory or planned production capacity "available to promise" to the customer on a regional or global basis. Many companies have both technical and business reasons for adopting enterprise software.

Both small and large companies can benefit both technically and strategically from investments in enterprise systems. Generally, the needs and opportunities of small companies are a subset of those facing large companies. For example, in addition to problems stemming from unintegrated legacy applications, large companies (particularly those that have grown through acquisition or have had a highly decentralized IT management regime) may also have the headaches of maintaining many different systems of the same application type. In large companies, it is not unheard of to find, say, 42 different general ledger packages or 22 separate purchasing applications in use at the time of adopting an enterprise system. Boeing, for example, had 14 bill-of-material systems and 30 shop-floor control systems before the company adopted ERP (Schneider, 1999). (See Table 10.1 for a summary of reasons that companies give for adopting enterprise systems.)

TABLE 10.1 Reasons for Adopting Enterprise Systems

	Small Companies/ Simple Structures	Large Companies/ Complex Structures
Technical reasons	• Solve Y2K and similar problems • Integrate applications cross-functionally • Replace hard-to-maintain interfaces • Reduce software maintenance burden through outsourcing • Eliminate redundant data entry and concomitant errors and difficulty analyzing data • Improve IT architecture • Ease technology capacity constraints • Decrease computer operating costs	Most small/simple company reasons plus • Consolidate multiple different systems of the same type (e.g., general ledger packages)
Business reasons	• Accommodate business growth • Acquire multilanguage and multicurrency IT support • Improve informal and/or inefficient business processes • Clean up data and records through standardization • Reduce business operating and administrative expenses • Reduce inventory carrying costs and stockouts • Eliminate delays and errors in filling customers' orders for merged businesses	Most small/simple company reasons plus • Provide integrated IT support • Standardize different numbering, naming, and coding schemes • Standardize procedures across different locations • Present a single face to the customer • Acquire worldwide "available to promise" capability • Streamline financial consolidations • Improve companywide decision support

It is important to take into account these wide variations in motivation to adopt enterprise systems when attempting to assess or explain their impacts and downstream consequences. Some goals are much more ambitious than others; if companies are like people, those with more ambitious goals are likely to achieve more than companies with less ambitious goals, but they are less likely to realize their full aspirations, and they may encounter many more difficulties along the way. Furthermore, enterprise systems may be better suited to realizing some goals than others. For example, the largest companies may face technical capacity constraints that prevent full data integration. Clearly, what companies think they are about when they adopt enterprise systems must figure somehow in the ways they approach the enterprise system experience and in the outcomes they achieve.

REASONS FOR NOT ADOPTING ENTERPRISE SYSTEMS

Of course, not all organizations adopt enterprise systems, even when they have some or all of the listed motivations for adopting. Some who do adopt enterprise

systems choose to use only certain modules, relying on legacy systems or new custom development for their remaining needs. And some who do adopt discontinue implementing or using these systems for a variety of reasons.[21]

One reason for nonadoption, partial adoption, or discontinuance is *lack of "feature-function fit"* between the company's needs and the packages available in the marketplace. "There are very few companies that don't have specialized processes dictated by their industry," according to one consultant (Slater, 1999). Many ERP system manufacturing modules were developed for discrete part manufacturing; these systems do not support some processes in process industries (e.g., food, paper), project industries (e.g., aerospace), or industries manufacturing goods with dimensionality, such as clothing and footwear. When organizational size and scale of operations are taken into account, there simply may be no commercially available package suitable for a particular organization.

More commonly, the organization may choose to adopt only certain parts of an enterprise system or may modify the system to improve feature-function fit. Consider examples of the implementation of SAP R/3 from Visio (a software company) (Koch, 1997). The first example concerns "deferred channel revenue." The article implies that Visio met this need with legacy code.

> Many software companies "dump" extra product with distributors at the close of a quarter so that they can pump up weak sales revenue totals. The downside of the strategy is that some of the extra software may flood back to the company in unsold returns. . . . To break the cycle, Visio tracks sales through to the retail outlets and compares the retail sales with the number of units shipped to distributors each month. If the retail stores sell less than Visio anticipated, Visio defers some of the revenues from the sales . . . ; if sales are up, it adds back some deferred revenues from previous months. . . . Unfortunately for Myrick and the Visio team, it's a complex and fairly unique way of handling revenues—two attributes that really annoy R/3. (Koch, 1997)

The second situation in which SAP R/3 did not meet Visio's needs concerns Visio's method of handling inventory.

> Visio outsources its manufacturing. . . . But R/3 doesn't let companies track something they don't own outright, and it doesn't recognize inventory that has no assigned value, like trial software, marketing handouts and other freebies. The consultants offer Visio two massively unpopular choices: Visio could assume ownership of the inventory throughout the manufacturing cycle, or it could send two invoices . . . [to customers]. . . . [The consultant] agrees to absorb the cost of fixing the inventory ownership process, ultimately conceding that sending two invoices to customers each month was unacceptable. (Koch, 1997)

The consultants apparently found a way to change the process with a bolt-on that did not require expensive and risky modification of the SAP code itself.

In addition to the lack of feature-function fit, a second major set of reasons for nonadoption, partial adoption, or discontinuance of enterprise systems concerns company growth, strategic flexibility, and decentralized decision-making style. Dell Computer Corp., for instance, planned full implementation of SAP R/3 but stopped after the HR module. The CIO claimed that the software would not be able to keep pace with Dell's extraordinary growth rate (Slater, 1999). Visio cited strategic flexibility as a reason for performing sales commission analysis outside of its enterprise system:

> "I wanted to retain the flexibility to change the commission structure when I needed to because it's such a critical process. It was my understanding that it might take awhile to do that within SAP and that once it was done, it wouldn't be so easy to change."

> Because the commission structure is so closely linked to the organizational structure of the sales groups, Buckley and Myrick decided to keep commissions analysis out of R/3. (Koch, 1997)

Companies that continually change their organizational structures and business models and particularly those that are not run "in a very top-down manner" may find enterprise systems unsuitable as a corporate solution (Bancroft, Seip, & Sprengel, 1997).[22] For example, at Kraft Foods Inc., a highly decentralized company that is gradually moving toward a "one-company" philosophy, enterprise systems were viewed as a culturally inappropriate strategy for systems integration (Bashein & Markus, 2000).

A third factor in the nonadoption of enterprise systems is the availability of alternatives for increasing the level of systems integration. Data warehousing, a bundle of technologies that integrates data from multiple source systems for query and analysis, provides what some describe as the "poor man's ERP." The usefulness of data warehousing as an integration strategy is limited by the quality of the source systems. Nevertheless, it can provide enormous relief for some organizations suffering from some of the technical problems shown in Table 10.1. Data warehousing was the integration strategy favored by Kraft Foods Inc. (Bashein & Markus, 2000).

Another alternative to enterprise systems involves rearchitecting in-house systems around a layer of middleware that isolates application systems from stores of "master data." When Dell abandoned SAP R/3 as its integration strategy, the company "designed a flexible middleware architecture to allow the company to add or subtract applications quickly and selected software from a variety of vendors . . . to handle finance and manufacturing functions" (Slater, 1999). Consultants say that rearchitecting systems with middleware is a viable alternative to enterprise systems when a company is basically satisfied with its software functionality and wants only to improve software integration and upgrade the user interface. This strategy is widely adopted in the financial services industries, where enterprise systems have made relatively few inroads (other than for administrative systems).

Discussions of reasons for not adopting enterprise systems usually conflate the issues just mentioned above with three other issues—cost, competitive advantage, and resistance to change. For example, Allied Waste recently announced plans to "pull the plug" on a $130 million computer system (Bailey, 1999). The reason given was that SAP was "too expensive and too complicated to operate." Yet it also seems clear that the software no longer fits the management style and structure of the company, as it presumably did at the time the decision to adopt SAP was made. Apparently, Allied Waste grew rapidly in the 1970s and 1980s when the firm was acquiring hundreds of trash haulers. When industry profits suffered, the company responded by cost cutting through centralized operations, a style of management well supported by SAP. Today, the company is moving toward much greater management decentralization. In this case, cost reasons, it seems, are tightly bound up with issues of management culture.

Some analysts have cited competitive advantage as a major reason for not implementing enterprise software (Davenport, 1998). Here, too, it is difficult to disentangle competitive advantage from explanations based in fit. If a company claims it will lose competitive advantage from adopting an enterprise system, it is also saying that it does things differently than the enterprise system does. The question here is whether lack of fit reflects an organization's "value-adding best practice" (albeit different from best practices in the software) or a costly inefficiency that the organization

is culturally unwilling to give up. Not surprising, vendors are more likely to cite "resistance to change" (or "lack of top management commitment") than they are to cite "competitive advantage" or "lack of feature-function fit" as a major reason for nonadoption of enterprise systems. In practice, careful analysis is necessary to determine whether or not an enterprise system is a good solution in a particular situation—and what the scope of the implementation project should be.

RECAP—IMPORTANCE OF ENTERPRISE SYSTEMS

Enterprise systems are an important topic for IS research for several reasons.

1. *Financial Costs and Risks.* Installing an enterprise system is an expensive and risky venture. Large companies have been spending on the order of hundreds of millions of dollars to make the technical and business changes associated with enterprise systems. There have been several visible enterprise systems failures, and nonacademic studies have questioned the financial and business payoffs from enterprise system projects. Therefore, enterprise systems raise questions that have long been studied in the IS field under the following labels: the payoffs from investment in information technology,[23] IS project success and failure,[24] and IS implementation process and change management[25] (training,[26] user involvement, communication, etc.).

2. *Technical Issues.* Enterprise systems are technically challenging. Among the more important technical areas of research around enterprise systems are the "development" life cycle for enterprise system packages; software selection approaches; enterprise modeling and software configuration tools and techniques; "reference models" for particular industry segments, systems integration strategies, and systems and software architectures; and data quality, reporting, and decision support for enterprise systems.

3. *Managerial Issues.* Enterprise system projects are managerially challenging since they may involve parties from many different organizations and cut across the political structures of the organization. Furthermore, enterprise systems have important implications for how companies should organize and manage their information systems functions. Finally, enterprise systems raise interesting challenges in terms of personnel and skill acquisition and retention. Therefore, the following areas of research are invoked by enterprise systems: IT project management;[27] IT project sponsorship and user involvement; IS-business relationships, vendor management, structuring the IS function and IT management more generally, and IS personnel management.[28]

4. *IT Adoption, Use, and Impacts.* Enterprise systems have been widely adopted across organizations, and the adoption of these technologies may spread further. However, it is not yet known how widely these technologies have been *assimilated* (Fichman & Kemerer, 1997)[29] in organizations, for example, how extensively they are used within the organization, how faithfully they are used, and how effectively they are used. Furthermore, these systems have large potential impacts[30] at all levels of analysis: individual and societal (employment, occupational structure, skills required, and quality of work), work system (cooperation, business process efficiency), organizational (competitive advantage, business results), and interorganizational (impact on supply chain, industry structure). For example, at the individual level, enterprise systems may entail a substantial increase in the visibility of an individual's performance,

leading to changes in the accountability and control regimes in the organization. In addition to the general topics of IT adoption, use, and impacts, enterprise systems are linked to research on business process reengineering, interorganizational information systems, and the strategic use of information technology.

5. *Integration.* Finally, enterprise systems suggest some unique questions not easily subsumable within existing bodies of information systems research. First, to what extent are enterprise systems bound up in a complete restructuring of organizations and industries around the capabilities of information technologies? Second, to what extent are we observing a structural change in the provision of IT services, from predominantly in-house to predominantly outsourced? What is the emerging role of the so-called system integrators (such as Andersen Consulting and IBM)? How should organizations manage a long-term IT development trajectory involving heavy dependence on external products and service companies? Third, what are the pros and cons of the enterprise systems approach vis-à-vis other strategies for achieving integration around information technology, such as rearchitecting systems around middleware and the object development paradigm or the looser integration strategy implied by data warehousing?

In short, the enterprise system phenomenon has strong conceptual links with just about every major area of information systems research. In addition, the phenomenon suggests the potential value of entirely new research directions.

The enterprise system phenomenon is so all-encompassing for organizations and their key business partners that it virtually demands a framework by which to understand it as a whole. As suggested, many bodies of literature and, hence, many theories are relevant to understanding important *pieces* of the enterprise system phenomenon, but what is lacking is an overarching framework within which many specific questions can be asked and their answers integrated. Our purpose in the remainder of this chapter is to propose such a framework. The framework is designed around the perspective of an enterprise system–adopting organization.[31] That is, the framework is designed to shed light on the questions facing the executive leadership of an organization considering whether, why, and how to participate in the enterprise system experience and what to do at various points in the process.

FRAMING THE ANALYSIS OF ENTERPRISE SYSTEM SUCCESS

This section sets up the description of our framework. We first address the questions the framework is designed to answer: What is success with enterprise systems? Why does success not always occur? What can be done to improve the chances of success? In other words, success is the key outcome of interest in our framework. In academic terms, it is our dependent variable.[32] More specifically, we define the success outcome as a multidimensional concept, a dynamic concept, and a relative one (to the concept of "optimal success," representing the best an organization can hope to achieve with enterprise systems). Next, we briefly review several broad categories of theoretical perspectives that purport to explain how and why enterprise systems success occurs. Of three categories of theories, we choose the "emergent process" type and, in particular, a specific theory of the business value of information technology as the basis for our framework. In the following section, we develop the framework more fully.

THE IMPORTANT OUTCOME: SUCCESS WITH ENTERPRISE SYSTEMS

> KPMG Management Consulting's recent report *Profit-Focused Software Package Implementation* showed some worrying results. Eighty-nine per cent of respondent companies claimed that their projects were successful, but only a quarter had actually obtained and quantified all the planned benefits. (KPMG, 1998)

To us, this quotation illustrates a fundamental gap in both practical and academic thinking about information systems—lack of consensus and clarity about the meaning of "success" where information systems are concerned. People rarely define terms such as success and failure; when they do, they invite endless complaints. For instance, one article in the trade press criticized a nonacademic study of enterprise system success for defining a project as a failure if project goals changed within a year (Ferranti, 1998).[33] Furthermore, people legitimately use a number of different definitions of success. As the KPMG quotation suggests, one can define success in terms of the implementation *project* (did the company succeed in getting the system up and running within some reasonable budget and schedule?) or in terms of *business results* (did the company succeed in realizing its business goals for the project?). The same ambiguity is clear in academic writing on the topics of information systems success and/or effectiveness. For example, one of the few academic studies of an enterprise system experience to date shows that success can look very different when examined at different points in time, on different dimensions, or from different points of view (Larsen & Myers, 1997).[34]

Our purpose is not to argue that one definition of success is inherently superior to another. Instead, we are claiming, first, that the key questions about enterprise systems *from the perspective of an adopting organization's executive leadership* are questions about success, for example: Will our investment pay off? Did our investment pay off? How should we go about implementing enterprise systems to achieve our goals? What can we do to increase the chances for success and avoid the risk of failure? Second, we are claiming that *no one measure* of enterprise system success is sufficient for all the concerns an organization's executives might have about the enterprise system experience. Instead, enterprise systems–adopting organizations require a "balanced scorecard" of success metrics addressing different dimensions (financial, technical, human) at different points in time. Based on our observations of enterprise systems projects, we argue that a minimum set of success metrics includes the following:

- *Project Metrics.* Performance of the enterprise system *project team* against planned schedule, budget, and functional scope. These are the classic performance measures applied to project managers.
- *Early Operational Metrics.* How *business operations* perform in the period after the system becomes operational until "normal operation" is achieved. Specifically, these metrics include some normally used to track the business as well as some unique to enterprise systems. Examples include labor costs, time required to fill an order, customer calls unanswered, partial orders filled, orders shipped with errors, inventory levels, and so on. Although the "shakedown phase" of an information systems project is transitional by definition, it is critically important for the implementing organization for several reasons, some of which have been well documented by others (Tyre & Orlikowski, 1994). When the business performs very poorly during the shakedown phase, the organization may lose business, sometimes

permanently, when the organization has yet to experience any major benefits to offset the large up-front investment. Exceedingly poor performance can lead to internal or external pressures to disinstall the system and in extreme cases can tip the organization into bankruptcy, as happened to Fox-Meyer Drug (Bulkeley, 1996). Consequently, organizations should include early operational performance in their definition of enterprise system success and should aggressively manage to minimize early operational problems and to resolve them quickly.

• *Longer-Term Business Results.* How the *organization* performs at various times after normal business operation has been achieved. Examples of relevant metrics include return on investment, achievement of qualitative goals such as "one face to the customer," better management decision making attributable to higher-quality data, continuous improvement of business metrics after operations return to normal, maintenance of internal enterprise system competence (among both IT specialists and end users), ease of upgrading to later versions of the enterprise system software, and so on.[35]

The foregoing discussion makes it clear that the success (or failure) of enterprise systems is not a monolithic concept. Rather, it is multidimensional and relative. It is relative, first, to the time at which it is assessed. For example, in our conversations with executives and integrators, we identified some companies with disastrous project and shakedown metrics but high levels of subsequent business benefits from enterprise systems. Conversely, we found companies with acceptable project and shakedown metrics that could not identify business benefits from installing the system. Similarly, an enterprise system that gives competitive advantage today may not do so tomorrow when competitors catch up and having such a system becomes a cost of doing business (McKenney et al., 1995).

Second, success is often judged relative to the organization's unique goals for the system. Two organizations with identical improvements in inventory carrying costs can be judged successful in different ways if the one's goals were to replace its legacy systems (more successful than expected) and the other's were to achieve an increase in market share (less successful than expected). At the same time, the company's goals, taken alone, make a poor standard against which to judge success. First, the company's goals may be insufficiently ambitious if they are compared to the inherent capabilities of enterprise systems and how well the organization needs to perform given its competitive position. For example, a company that is losing market share because it cannot promise delivery on a global basis would be "leaving money on the table" if it adopted an enterprise system solely to solve the Y2K problem and implemented it so that available-to-promise capability was not possible.[36] For another example, highly decentralized businesses may achieve less than is theoretically possible with enterprise systems if they configure the software so that each product business unit presents its own separate face to the customer. Conversely, the success of a company that achieved more with an enterprise system than it expected at the outset should be judged against a higher standard of performance than its unambitious goals. It might better be judged against the average business benefits realized by similar firms in its industry.

To accommodate the multidimensionality and relativity of enterprise system success from the adopting organization's perspective, we define a standard of *optimal success,* which refers to the best outcomes the organization *could* achieve with enterprise systems, given its business situation, measured against a portfolio of project, early operational, and longer-term business results metrics. Optimal success

can be *far more or less* than the organization's goals for an enterprise system. Furthermore, optimal success can be *dynamic;* what is possible for an organization to achieve may *change over time* as business conditions change. What we want the framework to help us predict or explain is an organization's *actual achievement* of an *enterprise system's success* (a scorecard of measures, assessed relative to optimal success—the best possible outcome). Looking ahead to the framework itself, we hope to show how and why an organization's decisions or actions at one point in time (e.g., during the "project") can result in optimal (or less than optimal) outcomes subsequently (e.g., during "shakedown").

We realize that organizations do not usually set out to achieve optimal success with information technologies; good enough is usually good enough. We also realize that optimal success is a theoretical abstraction that may be neither achievable in practice nor measurable in empirical research. Nevertheless, we believe the concept is theoretically useful because it "factors in" unintended positive and negative consequences of enterprise system adoption and organizational realities that are not fully reflected in the organization's enterprise system goals.

THEORETICAL PERSPECTIVES ON ENTERPRISE SYSTEM SUCCESS

Having defined the outcome of interest, we need a theory to explain it. An accepted classification scheme (Markus & Robey, 1988, derived from Kling, 1980, and Pfeffer, 1982) parses academic theories of IT-related outcomes into rational actor, external control, and emergent process theories. Rational actor theories emphasize the great, but bounded, ability of organizations and decision makers to realize their goals. An example of such a theory is the technology acceptance model (Davis, Bagozzi, & Warshaw, 1989),[37] which includes the factors that enter into an individual's choice of technology for particular tasks when faced with alternatives. On the plus side, rational actor theories highlight peoples' motivations and the actions they take to achieve their goals. Therefore, these theories tend to be very appealing to practitioners. A drawback of rational actor theories is that they downplay influential forces beyond the decision maker's control; furthermore, these theories accept managers' goals as givens without questioning their suitability relative to external constraints.

The second type of theory—external control theory—emphasizes the inexorable environmental forces, such as the trajectory of technological development or competitive industry forces. Academics employ external control theories when they claim that information technology alters industry structure or changes decision-making processes in an organization. A strength of external control theories is their explicit acknowledgment that organizations and people have less than perfect ability to make their goals a reality; on the downside, they minimize the ability of exceptional people and companies to change our world.

The third type of theory—emergent process—emphasizes the often unpredictable interactions between people in organizations and the environment. Emergent process theories assume that people try to achieve goals but acknowledge that the outcomes are often different from those intended—sometimes better and sometimes worse. External forces sometimes make a mockery of peoples' goals and actions, but sometimes the forces line up to favor the most unlikely goals. A prominent

example of an emergent process theory in the IS field is structuration theory (DeSanctis & Poole, 1994; Orlikowski & Robey, 1991). A strength of emergent process theories is that they account for mutual influences between the organization and its environment. Weaknesses include their greater explanatory than predictive power and the prominent role they assign to chance: Decision makers prefer prescriptive models that favor skill more than luck and that promise successful outcomes to those who follow the rules.

For a theory to be useful to practitioners, it must address their goals and "motivated behavior" (actions they can and do take with the intent to achieve their goals).[38] For a theory to fit the facts (not all companies are successful with enterprise systems, and not all the blame can be laid to their goals or actions), it must also address relevant factors that lie outside peoples' direct control. (For example, did the vendor actually deliver on the promise to release required functionality on a specified date? Is the software sufficiently bug free to support reliable operations?) Because emergent process theories combine goals and actions with external forces and chance, we chose this theoretical structure for modeling the enterprise system experience.

More specifically, we build our framework on a particular emergent process theory designed by Soh and Markus (1995) to explain how information technology creates (or fails to create) business value.[39] This theory contributes three key points to an understanding of the success of enterprise systems. First, it argues that the necessary conditions for a successful outcome (in their model, high-quality information technology "assets") are not always sufficient for success. Occasionally, an IT investment on track for success is derailed by an external event (e.g., competitors' responses) or changing external conditions (e.g., recession). Chance and randomness can play an important role in the outcomes achieved.

Second, the Soh and Markus (1995) framework describes the "IT investment to business value" process as a series of three linked models that correspond to the phases of a typical IT investment, roughly speaking, system development, implementation, and ongoing operation. (See Figure 10.1.) The outcomes of one phase became starting conditions for the next. Thus, decisions and actions in a phase may increase or decrease the potential for success ("optimal success") subsequently. Furthermore, because each phase generally involves different groups of people, the framework directs attention to communication difficulties that accompany "the handoffs" from one phase to the next.

Third, the Soh and Markus (1995) framework explains the outcomes of each phase as resulting from interactions between external conditions and the activities that characterize the phase. This means that both uncontrollable events and choiceful human actions can influence outcomes. At the same time, events and actions may be unsynchronized, and delayed effects may result in unresolved problems (or even opportunities) requiring action down the road. Thus, problems can accumulate, cascading toward a wildly unsuccessful outcome.

Two important changes are needed to adopt the Soh and Markus (1995) framework to the enterprise systems experience. First, the outcome variables need to be changed from business value (and other intermediate outcomes) to success as defined earlier. Second, the framework requires the addition of an initial phase that includes the important organizational decisions and actions before the project officially starts. In many organizations, executives, technical specialists, and consultants can spend considerable time (and money) making decisions about whether,

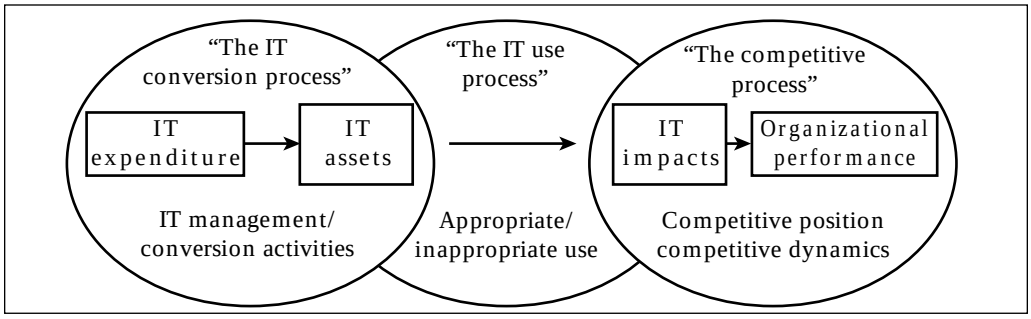

FIGURE 10.1 Soh and Markus (1995) Model

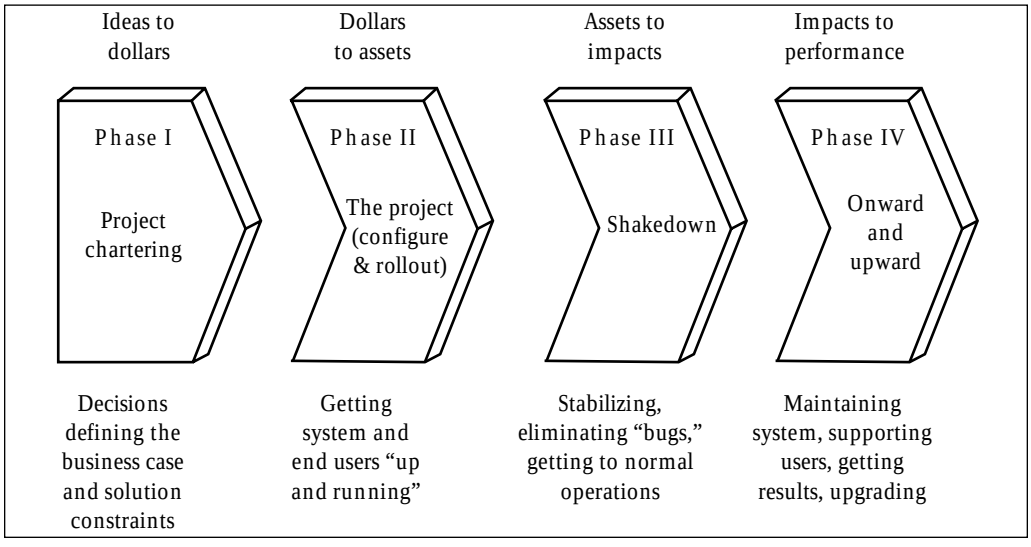

FIGURE 10.2 Enterprise System Experience Cycle

why, and how to undertake enterprise systems. For instance, any of the following may occur before a project manager is named and a budget is approved: documenting current business processes, analyzing the potential for improvement, comparing processes with the "reference models" or "best practices" embedded in enterprise systems software, selecting software, deciding which software modules to implement in what sequence, and deciding how to roll out the new functionality to various business units. Therefore, we identify a new first phase, which we call *chartering.* The remaining three phases in the enterprise system experience cycle (the *project phase,* the *shakedown phase,* and the *onward and upward phase*) correspond loosely to the three phases in the Soh and Markus model (see Figure 10.2).

In the next section, we describe our framework in greater depth.

THE DYNAMICS OF ENTERPRISE SYSTEM SUCCESS

An organization's experience with an enterprise system can be described as moving through several phases, characterized by key players, typical activities, characteristic problems, appropriate performance metrics, and a range of possible outcomes. Each enterprise system experience is unique, and experiences may differ considerably,

depending, for example, on whether the adoption of the enterprise system is initiated by IS specialists or by businesspeople, involves external consultants or is done largely in-house, follows a process of strategic IT business planning or business process reengineering or does not follow such a process, and so forth.

Table 10.2 summarizes the four "ideal" phases of the enterprise system experience cycle,[40] and the phases are briefly described in the following section. It is important to note that organizations recycle through the phases when they undertake major upgrades and/or replacements of their enterprise systems.

THE PHASES DESCRIBED

The Chartering Phase

The chartering phase comprises decisions leading up to the funding of an enterprise system. Key players in this phase include vendors, consultants, company executives, and IT specialists, although the precise constellation of players may vary. (Sometimes vendors sell directly to company executives, with minimal IT involvement; other times the decisions are driven by IT specialists, with minimal executive involvement.) Key activities include building a business case for enterprise systems, selecting a software package (though this decision may be deferred until the project phase), identifying a project manager, and approving a budget and schedule. A large number of errors or problems can arise during this phase. The business case for investing in an enterprise system can be incomplete or faulty; the organization may seriously underestimate the need for business and organizational change in conjunction with the software implementation; objectives and metrics for the initiative may be left undefined (Ross, 1999). The outcome of this phase may be a decision not to proceed with the enterprise system or a decision to proceed. If the latter, the chartering decisions passed on to the next phase may be sound or unsound. An example of an unsound charter is a build-to-order company purchasing an ERP package designed for a make-to-stock business (Slater, 1999). Another example is the decision not to allocate sufficient resources for change management and training (Ross, 1999). A third is the decision of a decentralized company to require more standardization of business processes than is necessary to achieve business benefits (Davenport, 1998). Still another is the choice of a highly inexperienced project manager.

The Project Phase

The project phase comprises activities intended to get the system up and running in one or more organizational units. Key players include the project manager, project team members (often nontechnical members of various business units and functional areas), internal IT specialists, vendors, and consultants. Again, the constellation will vary, depending on the decision to do the project in-house, with outside assistance, or on an outsourced basis. Key activities include software configuration, system integration, testing, data conversion, training, and rollout. Again, a large number of errors and problems can occur. Project teams may be staffed with inadequate representation; teams may lack requisite knowledge and skills; teams may embark on extensive, unnecessary modifications; data cleanup, testing, or training may be inadequate. In addition, of course, the business conditions characterizing the chartering phase may have changed: The company may have fallen into financial

TABLE 10.2 The Enterprise System Experience Cycle by Phase

Phase Name, Description, and Key Actors	Typical Activities	Common Errors or Problems	Typical Performance Metrics	Possible Outcomes

Chartering ("ideas to dollars")
• Decisions leading up to project approval and funding
• Executives, selected IT specialists, enterprise systems vendor, and/or consultants (may be IT driven with low executive involvement or executive driven with low in-house IT involvement)
	• Idea of adopting enterprise systems surfaced
	• Business case for investment developed (may be highly informal)
	• Definition of key performance indicators and process of measurement
	• Current state analysis (may be deferred or not done)
	• Selection of software, hardware platform, networking, database, implementation partner, project manager (may be partially or totally deferred to project phase)
	• Initial plans for how system will be rolled out, supported, and maintained, upgraded, etc. (may be deferred)
	• Communication to organization
	• Organizational changes and/or incentives related to enterprise system and/or organizational performance improvement, if any (may be deferred)
	• Decision to proceed, approval of project plan
		• Overselling by software vendors and implementation consultants
		• Failure to link technology plan to business strategic plan
		• Unrealistic business case and project parameters
		• Key performance indicators not or poorly defined, including the measurement process and ownership of this
		• Selection of inappropriate software, hardware, integrator, and/or project manager; inadequate contracting with external parties
		• Inadequate contracting with vendors and consultants
		• Lack of long-term support and migration strategy
		• Failure to recognize need for business change; underestimating change management difficulty
		• Misunderstanding organizational requirements, particularly as related to need for data access and reporting
			• Not usually formally measured
			• Possible metrics include quality of business case, fit with business strategy, relevance of key performance indicators, adequacy of schedule and budget, soundness of project parameters, and constraints
				• Enterprise systems idea abandoned as unlikely to provide business benefits
				• Decision to proceed with a project with certain parameters (schedule, scope, and budget)
				  • Business case for project is unsound, creating potential for problems later
				  • Business case for project is sound

(continued on the next page)

TABLE 10.2 *(continued)*

Phase Name, Description, and Key Actors	Typical Activities	Common Errors or Problems	Typical Performance Metrics	Possible Outcomes

Project—Configuration, integration, and rollout ("dollars to assets")
• Activities designed to get the system up and running in one or more organizational units
• Project manager, project team members, and a variety of external technical and general management consulting resources, executives (in steering committee capacity), other organizational members (in consultative roles)
	• Development of detailed project plan
	• Ongoing project management
	• Selection and assignment of project team members
	• Training of project team members and acquisition of supportive skills
	• Current and/or future business process modeling and reengineering, if any
	• Execution of change management plan, if any
	• Software configuration
	• Software customization, if any
	• System integration
	• Integration of software bolt-ons and/or legacy systems, if any
	• Data cleanup and conversion
	• Documentation
	• Testing, bug fixing, and rework
	• Executive and end-user training
	• Rollout and startup
		• Staffing project subteams without appropriate cross-functional representation
		• Difficulty acquiring adequate knowledge and skill in software configuration (especially cross-module integration), integration of bolt-ons or legacy systems, and a variety of platform technologies
		• Poor-quality software, documentation, training materials
		• Inadequate knowledge on part of consultants and vendor personnel
		• Configuring software for multiple units on the basis of analyzing only one unit
		• Assuming that end-user training should be funded from operations budgets
		• Configuration errors that require rework if caught
		• Customizations that do not work
		• Failure to manage project scope, schedule, budget
		• Inadequate attention to data cleanup
			• Project performance to schedule, scope, and budget	• Project terminated owing to overruns or intractable technical problems
				• Rollout of some operational enterprise system functionality to one or more organizational units
				  • Functionality, operational performance, and organizational preparation are insufficient to address business needs
				  • Functionality, operational performance, and organizational performance are sufficient to address business needs

TABLE 10.2 *(continued)*

Phase Name, Description, and Key Actors	Typical Activities	Common Errors or Problems	Typical Performance Metrics	Possible Outcomes
	• Inadequate attention to reporting • Short-cutting testing and/or training when schedule gets tight • Lack of technology transfer from external consultants • Vendor delivery and software performance problems • Lack of software ease of use • Turnover of project personnel			

Shakedown ("assets to impacts")
• Period of time from "going live" until "normal operation" or "routine use" has been achieved
• Operations managers, end users, remnants of the project team, IT support personnel, external technical support personnel
	• Bug fixing and rework
	• System performance tuning
	• Adding hardware capacity
	• Problem resolution
	• Process and procedure changes
	• Retraining, additional training
	• Adding people to accommodate learning and shakedown needs
		• Business disruption
		• Difficulty diagnosing and solving performance problems
		• Excessive dependence on "key users" (project team members) and/or IT specialists
		• Maintenance of old procedures or manual workarounds in lieu of learning the relevant system capabilities
		• Data input errors
		• Poor software ease of use
		• No growth of end-user skills after initial training
		• Underuse/nonuse of system
		• Failure to achieve normal operations ("system" never stabilizes)
			• Relevant system performance measures such as downtime, response time, etc.
			• Short-term changes in key performance indicators such as customer calls missed, labor costs, order fulfillment cycle time and error rates, length of time required to complete financial close
			• Short-term impacts on customers and suppliers
			• Employee job quality/stress
				• Project terminated owing to severe shakedown projects (e.g., disruption of business, poor technical performance, bugs and errors)
				• Normal operation with routine use of enterprise system is achieved
				  • Impacts insufficient to address business needs
				  • Impacts sufficient to address business needs

(continued on the next page)

TABLE 10.2 *(continued)*

Phase Name, Description, and Key Actors	Typical Activities	Common Errors or Problems	Typical Performance Metrics	Possible Outcomes

Onward and upward ("impacts to outcomes")
• Routine operation of the business until such time as a new version of enterprise system is rolled out
• Period during which business realizes business benefits from system, if any
• Operations managers, end users, IT support personnel, executives
	• Postimplementation investment audit (may not be done)
	• Continuous business improvement (may not be done)
	• Technology upgrading/migration (may not be done)
	• Additional end-user skill building (may not be done)
		• Not assessing system-related outcomes on a routine basis
		• Enterprise system of today becomes legacy system of tomorrow (organizational unwillingness or inability to make technology upgrades)
		• No available documentation on configuration rationale
		• Turnover of knowledgeable personnel (IT and end user)
		• No organizational learning about IT projects, enterprise systems
		• Failure to manage to the intended results of the enterprise system
			• Not usually formally measured
			• Possible indicators include continuous business performance improvement/learning metrics, end-user skill assessment, ease of upgrading/migration, shortening of project and shakedown phases over time, IT specialist competence measures
				• Unwillingness or inability to improve business performance and/or migrate technically (e.g., extreme dissatisfaction with implementation process or outcomes, loss of technical or end-user competence)
				• Formal or informal assessment that investment has been unsuccessful (e.g., failure to achieve hoped-for business results, unexpected costs, or consequences)
				• Formal or informal assessment that project has achieved goals and/or unexpected benefits
				  • Organization fails to improve its overall competitive position
				  • Organization improves its overall competitive position

distress, it may have merged with another company, or it may have shifted business models. Some projects are terminated owing to cost or schedule overruns or severe technical problems. Others result in the rollout of the operational enterprise system functionality to one or more organizational units. If the latter, the enterprise system functionality, operational performance, and organizational preparation may be sufficient to fit the organization's goals and/or needs, or they may be insufficient for "success."

The Shakedown Phase

The shakedown phase is the organization's coming to grips with the enterprise system. The phase can be said to end when "normal operations" have been achieved (or the organization gives up, disinstalling the system). The project (or consulting) team may continue its involvement or may pass control to operational managers and end users and whatever technical support it can muster. Activities include bug fixing and rework, system performance tuning, retraining, and staffing up to handle temporary inefficiencies. To a large extent, this is the phase in which the errors of prior phases are felt—in the form of reduced productivity or business disruption—but new errors can arise in this phase too. For example, operational personnel may adopt workarounds to cope with early problems and then fail to abandon them when the problems are resolved. Similarly, the organization may come to rely excessively on knowledgeable project team members rather than building the enterprise system knowledge and skills in all relevant operational personnel. As mentioned, some enterprise systems are terminated during the shakedown phase, as in the case of Fox-Meyer Drug (Bulkeley, 1996). Alterntively, organizations may achieve (or declare) "normal operations." If the latter, the impacts attributable to the organization's use of the system may fit its goals or business needs, or they may fail to do so.

The Onward and Upward Phase

The onward and upward phase continues from normal operation until the system is replaced with an upgrade or a different system. It is during this phase that the organization is finally able to ascertain the benefits (if any) of its investment. Key players include operational managers, end users, and IT support personnel (internal or external). Vendor personnel and consultants may also be involved, particularly when deliberations about upgrades are concerned. Characteristic activities of this phase include continuous business improvement, additional user skill building, and postimplementation benefit assessment; however, these "typical" activities are often not performed. A common problem of the onward and upward phase is the loss of knowledgeable personnel who understand the rationales for prior configuration choices and how to improve the business processes through the use of the system. Several ultimate outcomes are possible: The organization may be unwilling to undertake further improvements or upgrades. The organization may decide that its investment has been unsuccessful in meeting goals or business needs. Or the organization may decide its experience has been a success. If the latter, the organization's competitive position may or may not have been improved as a result of its use of enterprise systems.

TOWARD A PROCESS THEORY OF ENTERPRISE SYSTEM SUCCESS

Each enterprise system experience runs a different course, but across the variations, regularities can be found.

- Many different things can go wrong in each phase of the enterprise system experience cycle. Furthermore, not all problems or errors are immediately detectable (and, hence, they are not all immediately correctable).
- There are several possible outcomes for each phase. One is an "optimal" outcome, for example, in the chartering phase, the decision to proceed with an enterprise system project that has a sound business case. A second outcome is a "termination" outcome, such as the decision not to proceed with the enterprise system because analysis revealed an unacceptable business case. A third outcome might be called "continuation with undetected and uncorrected problems" or "unresolved experience risk."[41] The subsequent phase inherits these unresolved risks.
- This third outcome is analogous to what sociotechnical systems theorists call a "variance" (Taylor, 1993). Variances in production processes are deviations from standards in the inputs to a production activity (e.g., raw material quality defects). Variances are not necessarily detected right away; if they cause problems, they may do so only much later in the production process after much money has been expended in working the raw material. Similarly, requirements definition errors in software development may not show up until the system is put into production. Unresolved variances in each phase of an enterprise system experience are passed on to the next phase, where they may or may not be detected and appropriately resolved (depending on probabilistic processes). So, for example, some variances in the chartering phase may remain uncorrected until they show up in the onward and upward phase as a lack of business benefits. In general, the cost of fixing problems increases with delays in recognizing and correcting variances.
- Generally speaking, different actors are involved in different phases of the enterprise system experience cycle. While there may be some continuity across phases (for example, oversight by an executive steering committee during the project phase), handoffs to a different group of people (with different specialties, experiences, and skills) increase the likelihood that variances passed on from earlier phases will not be caught and resolved until they create significant problems. For example, project teams rarely catch and correct significant errors (e.g., failure to match the project to business strategy) in the business case that forms their "charter."
- Of course, not all variances end up causing problems and requiring fixing or rework. Whether or not variances cause problems depends on probabilistic processes such as bad luck, changing business conditions, interactions with other variances, and so on. For example, a badly configured enterprise system requiring expensive rework may not be a problem if the organization's financial position remains sound. Furthermore, it is possible for external conditions and the organization's decisions and actions to interact in such a way that the outcome is better than it was at a prior point, increasing the standard of optimal success. For example, successful implementation of ERP software, while perhaps not providing immediate business value to the adopting organization, might nevertheless position that organization to take advantage of supply-chain integration, thus improving its competitive position relative to competitors.

Two brief examples illustrate how our framework works. The first example concerns Quantum Corp.'s implementation of the Oracle suite of enterprise software applications (Radosevich, 1997). Consultants generally advise against "big-bang" implementations (implementing all modules at once and/or implementing in all locations at once) because of the high potential for business disruption. During its chartering phase, Quantum's leadership determined that acquiring the capability instantly to commit inventory to customers on a worldwide basis (called available to promise, or ATP) was a strategic necessity.

> Because the ATP functionality touches on several areas, including sales processing, inventory and logistics, Quantum had to have the entire suite operating before attaining its key business goal. (Radosevich, 1997)

Quantum therefore decided to implement everywhere at once, but the leadership clearly understood the "bet-your-business" nature of this decision should the shakedown phase not go as well as planned. Therefore, during the project phase, the implementation team made extraordinary efforts in testing the system. Furthermore, when early testing revealed problems, the rollout date, scheduled for summer 1995, was postponed. The rollout finally began as a planned shutdown worldwide:

> At the stroke of midnight on April 26, 1999, every business system at Quantum Corp's offices across the globe went down. . . . The systems stayed down for a week. . . . On May 5, Quantum switched on the new system, and it has been running successfully ever since. (Radosevich, 1997)

In this example, the organization deliberately made a decision involving a high level of risk but took appropriate steps to ensure that the risks did not materialize into a disruption that might have threatened the existence of the organization.

Contrast this example with that of Fox-Meyer Drug (Bulkeley, 1996). When this SAP implementation was chartered, the CIO at least was aware that it was a "bet-your-business" proposition. Yet the $60 million project was approved at the same time that the company also embarked on a state-of-the art $18 million automated warehouse. During the project phase, bad luck intervened: Fox-Meyer Drug lost a large customer, accounting for 15% of its business. To increase revenues, the company aggressively bid on new business: It figured contract pricing on the assumption that the projected annual $40 million savings from the SAP project would be realized immediately on startup, and it decided to advance the SAP rollout by 90 days. That close to the end of the project, little was left to do other than training and testing. So project team members decided not to test modules that had not been customized (thus failing to detect configuration errors). Cutover to the new system resulted in disaster. Meanwhile, the automated warehouse also did not perform as planned. It was estimated that the company sustained an unrecoverable loss of $15 million from erroneous shipments. The company was forced into bankrupcy, and shareholders have since sued both the enterprise system vendor and the integration consultant for $500 million each.

In this example, the chartering decisions were moderately risky. When bad luck occurred during the project phase, the company's decisions had the effect of increasing rather than decreasing risk. When major problems finally materialized during shakedown, the organization did not have the time or the resources to overcome them.

FACTORS IN ENTERPRISE SYSTEM SUCCESS

One can abstract from such examples and the details of Table 10.2 a more general theory of success in the enterprise system experience. At any one moment in time (phase), an enterprise system–adopting organization faces a situation that involves conditions and events (some of them outside its direct control) with an ability to make plans and take actions (that is, goal-directed or "motivated" behavior). These elements of the situation are the factors in (influences on) the outcomes that become inputs at the next moment in time (phase). In the next section we briefly describe these success/failure factors.

Factors External to an Organization's Control

The organization adopting an enterprise system faces several *starting conditions* such as competitive position, industry, financial position, prior relevant experience, size, structure, and management systems that may predispose it to success or failure. While there are undoubtedly threshhold levels for some of these conditions, they generally can not be said to be necessary (or sufficient) for the success of the enterprise system, since organizations have been known to succeed or fail despite them. But these factors come into play in the enterprise system experience in two ways.

First, organizations' goals and plans for enterprise systems may or may not be realistic when viewed objectively in light of these conditions. Dell, for example, decided (after some experience) that an enterprise system was not sufficiently flexible for its rapid growth. For another example, an organization on the brink of bankruptcy may not have enough time and money to realize the benefits of an enterprise system. Starting conditions define the needs and opportunities of organizations relative to enterprise systems (whether or not organizations recognize them for what they are).

Second, starting conditions may not remain the same over the course of the enterprise system experience. After a company decides to customize the enterprise system software, the vendor delivers the needed functionality. After the company has configured the enterprise system for a particular way of doing business, the company merges or sells off a major line of business. Sometimes changes in conditions favor the organization's plans. But probably more often, changing business conditions derail plans. Successful organizations modify goals, plans, and execution to bring their behavior back into line with the environment.

Organization's Motivated Behavior

The organization's goal-directed enterprise system behavior can be defined in four categories: goals, plans, execution, and responses to unforeseen problems. First are the *goals* themselves. Some goals are more conducive to success than others, some are too unambitious to be motivating, and others are unrealistic in light of the objective characteristics of the enterprise system and the organization's starting conditions. Given the great complexity and expense of enterprise systems, for example, some analysts argue that only companies seeking to streamline business processes, to standardize data, or to standardize processes can achieve a positive return on their enterprise system investment (Connolly, 1999).

Plans are another factor in the equation. Plans (and policies) such as not to customize, to reengineer first (last, or not at all), and to phase the rollout are essential to keeping the project phase on track. Enterprise system integrators often claim

to have "*the* methodology" that will guarantee success, but not all plans are created equal. The organization's plans for an enterprise system must be linked to its starting conditions and goals. Traditional organizations may need much more change management activity than those in the volatile high-tech sector. The need for a particular business capability may necessitate a risky big-bang rollout (Radosevich, 1997). For another example, the tier-one supplier to an automotive assembler should be much more concerned than a tier-two supplier about the potential negative impacts in lost business from plans that shortchange testing.

The best laid plans are worthless if they are not followed. Good *execution* is something that a consultant's methodology cannot guarantee. If configuration tasks exceed the schedule, cutting the time allotted to testing and training may not guarantee failure, but, given these choices, success will require more than a little luck.

No matter how well an organization executes plans well designed to meet its carefully thought-out goals, conditions may change and unforeseen problems may arise. Successful organizations successfully *resolve* problems by changing their goals, plans, and actions to get a favorable outcome.

Intermediate and Final Outcomes

Starting conditions, changes in conditions, goals, plans, and actions interact (Orlikowski, 1996). Resulting from these interactions are unresolved risks and problems (as well as opportunities, although avoiding failure is usually the primary concern). Unresolved risks and problems themselves interact with changing business conditions and the organization's actions in response to them. If the experience is not terminated, the interactions in one phase result in starting conditions for the next. In economic terms, the course of the enterprise system experience exhibits "path dependence." The final outcome may be very close to optimal success (itself a moving target) or suboptimal on one or more dimensions.

Graphically, the abstract theory described here is depicted in Figure 10.3 for a single phase (negative situation only). Table 10.3 summarizes the details of the process theory as it applies to the specific case of enterprise systems. Table 10.3 is organized according to the important dimensions of process models (Soh & Markus, 1995) by phase: outcome, necessary conditions, probabilistic processes, and recipe for success.

USES OF THE FRAMEWORK

This simple theory of the enterprise system experience has several benefits and uses. First, it is framed in terms that are meaningful to practitioners, but it avoids simplistic overemphasis on a single factor (such as methodology). Furthermore, it explicitly recognizes the role of factors outside the organization's direct control, thereby focusing attention on both planning and the resolution of unforeseen problems. Perhaps most important, it emphasizes the importance of goals—something that technical specialists in particular often take as givens. In view of their greater knowledge of system capabilities and limitations, technical specialists can increase the probability of success by (tactfully) challenging the organization's goals for the enterprise system and these goals' fit with business strategy and external conditions.

As noted by Silver and colleagues (Silver et al., 1995), the theory outlined here can be used both retrospectively and prospectively. Retrospectively, it is useful for

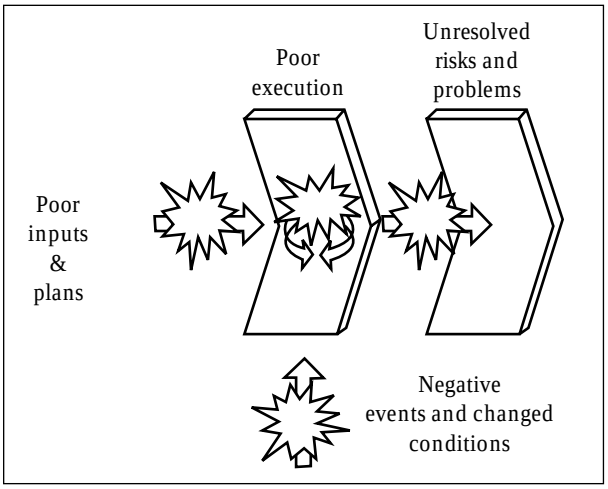

FIGURE 10.3 Factors in Suboptimal Success

tracing back problems and suboptimal success at each phase to variances arising in earlier phases. For example, configuration errors (which disrupt business operations during shakedown) can result from chartering phase errors (selection of inappropriate goals, software, or partners) as well as from project phase problems (poor composition of project team, inadequate change management execution). Prospectively, the theory can be used to identify a large number of potential problems that should be addressed in basic and contingency plans. It can also help sensitize decision makers to the need to change plans and actions in light of common problems or changing business conditions.

DIRECTIONS FOR FUTURE RESEARCH

The theoretical framework outlined in this theory is too broad in scope for direct empirical testing. However, many lines of research follow directly from the framework, and the phenomenon raises some research themes and issues beyond the limits of the framework.

RESEARCH ON THE PROCESS THEORY

The basic structure of the framework is a sequence of phases, each with intermediate outcomes. The intermediate outcomes are argued to influence the final outcome, but not to determine it; in other words, events and actions taken at any point in the experience cycle may derail an experience that appears to be destined for success or may help a troubled experience get back on track. This general proposition lends itself to testing via multiple methods, including multiple case study and a survey approach. One wants to know the proportion of successes at each phase that are successful in the next and what kinds of actions and events are most likely to change the course of an experience. An important issue concerns the specific metrics of success. Which metrics in each phase have the greatest predictive and explanatory power?

The framework gives a special role to the decisions of the chartering phase. Since errors in chartering an enterprise system experience may not be caught and

TABLE 10.3 A Process Theory of Enterprise System Success

Phase	Successful Outcome	Necessary Conditions	Probabilistic Processes	Recipe for Success
Chartering ("ideas to dollars")	Decision to proceed with enterprise systems in a properly chartered project with a sound business case	• Executive participation • Sound assessment of business conditions and needs • Good understanding of enterprise systems' capabilities and limitations • Carefully constructed business case well communicated to relevant parties	• Managerial decision making ("garbage can" dynamics and politics) • Distribution of good information about environment and technology • Availability of people willing and able to challenge untested assumptions • Human communication gaps and acceptance of need to change • Volatility of business conditions	Success occurs when executives make sound decisions about investing in enterprise systems and bring the organization into alignment with these decisions
Project ("dollars to assets")	Rollout, within reasonable cost and schedule, of enterprise systems functionality that is operational and sufficient to address business needs to an organization prepared to accept it	• Expenditures • Participation by various organizational groups • Technical resources, methodologies, and expertise • Project, vendor, and stakeholder management • Organizational change management expertise • Knowledgeable and skilled IT specialists and project participants	• Quality of business plan resulting from earlier phase • Volatility of business conditions • Availability and quality of technical resources, methodologies, expertise • Execution of project plan • Stakeholder politics • Resolution of problems arising during project phase	Success occurs when 1. The project team faithfully executes a sound project plan and appropriately responds to technical and human challenges that arise during the project OR 2. The project team appropriately modifies the project plan to match changing business and organizational conditions
Shakeout ("assets to impacts")	Normal operations achieved within reasonable time frame and expense with impacts that are sufficient to meet business needs	• Trained users • Well-configured and integrated enterprise system • Redesigned business processes • Additional human, financial, and technical resources to cope with problems arising during shakeout	• Quality of assets resulting from earlier phases • Volatility of business conditions • Stakeholder politics • Execution of shakedown phase problem resolution activities	Success occurs when 1. The organization is well prepared to accept and use a system and related infrastructure of sufficient quality to meet business needs OR 2. Appropriate measures are taken to fix technical and organizational problems arising during shakedown quickly and effectively

(continued on the next page)

TABLE 10.3 *(continued)*

Phase	Successful Outcome	Necessary Conditions	Probabilistic Processes	Recipe for Success
Onward and upward ("impacts to outcomes")	Organization improves its competitive position as a result of enterprise system project and maintains its technological and business flexibility for future developments	• Managers committed to achieving business results (e.g., to use system-generated information to improve organizational performance) • Impacts attributable to use of high-quality enterprise system and infrastructure • System, technical infrastructure, business processes, and human resources sufficiently flexible to adapt to changing business conditions • Adequate resources devoted to maintaining and renewing system, technical infrastructure, and human competence	• Quality of assets resulting from earlier phases • Volatility of business conditions • Stakeholder politics • Execution of results management and maintenance/enhancement activities • Survival of software vendor	Success occurs when 1. Benefits from use of the system combine with favorable competitive conditions AND 2. The future evolution of the enterprise system and related infrastructure is well managed

corrected during the project phase, they are likely to prove expensive to correct later, if they are not outright irreversible. Empirical research is needed to address the chartering phase errors that are most likely, most consequential, and most difficult to detect. Why do these errors occur? What can be done to prevent or correct them?

Adopting organizations with many subunits face chartering and project complexities that single-location businesses do not. Many management theories assume that certain "configurations" of strategy or structural variables are better suited than others to the contingencies of particular environments. The same may be true of the strategies that companies develop to deal with enterprise systems, particularly in complex organizations with many subunits. One wants to know first of all whether there are relatively stable clusters of chartering decisions (e.g., big-bang rollout coupled with no local autonomy on software configuration). If they do exist, how do organizations decide which strategic configurations to adopt? Are these strategic configurations of decisions more successful in the enterprise system experience if they fit well with the various aspects of their environment such as industry type, organizational structure and business model, organizational culture, and the like?

An especially interesting part of the model is what happens after "normal operation" is achieved. When, how, and why does the organization decide to keep things as they are, making the enterprise system of today into the legacy system of tomorrow? When, how, and why do organizations upgrade? Is it better to upgrade frequently so as not to risk big conversions? Do organizations really perceive themselves to be locked in to a particular vendor? How does this influence their subsequent behavior?

Finally, an underlying theme in the framework is the role played by human knowledge and skill—and *gaps* in knowledge and skills—in enterprise system success. What are the relevant bodies of knowledge? How are they distributed externally (across vendors and consultants) and internally (across IT specialists, executives, and users)? Under what conditions are they transferred effectively? What are the barriers to effective knowledge transfer? How can organizations acquire and maintain the relevant expertise? These are just a few of the knowledge management questions posed by enterprise systems. One potentially significant knowledge issue concerns the loss of knowledge about enterprise system configuration (e.g., through poor documentation and personnel turnover). One system integrator told us he believed that it might be *more* costly to update a poorly documented enterprise system than a comparable legacy system because with the enterprise system, the requisite knowledge most likely lies outside the organization. This is an interesting testable hypothesis that gets at the heart of the advantages claimed for packages over in-house development.

BEYOND THE FRAMEWORK

In addition to the many questions raised by the framework itself, the phenomenon of enterprise systems raises interesting questions that go well beyond the framework.

- *Dependence on Vendors.* Because they are so all-encompassing, enterprise systems create a level of dependence on a single software vendor that far surpasses the dependence associated with prior technological regimes. Does this dependence have negative effects on organizations? How do the effects manifest themselves?

How do organizations cope? What are the costs of picking the "wrong" vendor (one that goes out of business or lacks the resources for continued product enhancement in the company's industry segment)?

- *Uniqueness.* To what extent do the enterprise system experience cycle and related process theory apply more generally to other types of information systems? What is truly unique and different about enterprise systems, as opposed, say, to software environments such as Lotus Notes or to homegrown applications?
- *Other Kinds of Integration.* Some organizations have eschewed enterprise systems for any number of reasons. Nevertheless, many of the motivations for enterprise systems remain, particularly in large, complex organizations. To what extent are these organizations pursuing alternatives to enterprise systems as a form of internal integration? (Alternatives might include summary-level integration via data warehousing, rearchitecting legacy systems with middleware, or redevelopment using the object paradigm.) What are the advantages and disadvantages of different integration approaches? What are the consequences of these alternative choices for organizational performance and future flexibility? In particular, are there observable negative consequences from the "tight coupling" entailed with enterprise systems?[42]
- *The "Extended Enterprise."* The enterprise system experience framework assumes that the organization is the most appropriate level of analysis. Increasingly, however, organizations are looking to extensions of enterprise systems to connect themselves more tightly with customers and suppliers in what is often called "supply-chain integration." Suppliers manage customers' inventory, and redundancies are eliminated across organizational lines instead of just within them. Software enables each party to access the other's data—perhaps it is more appropriate to speak of interorganizationally shared software and data resources. To what extent does the enterprise system experience framework apply to the interorganizational integration experience? Can the framework be extended, or is an entirely new framework needed? Finally, of course, what are the societal consequences of this trend, if it continues?
- *Other Kinds of Coordination.* Organizations for which supply-chain integration is not a valid option may still need interorganizational coordination that cannot be supplied by market mechanisms (King, 1999). What kinds of information technology–mediated mechanisms will industries such as health care use for coordination? How will the costs and benefits of "nonintegrated coordination" (King, 1999) differ from those of supply-chain integration and electronic markets?
- *Influences on Vendors.* How do enterprise system–adopting organizations influence the strategic development plans of enterprise system vendors? Which adopters are influential and how do they exert their influence? What are the other influences on vendors' behavior (e.g., shareholders, media, technology marketing research firms)?

CONCLUSION

Enterprise systems represent an important contemporary phenomenon in the organizational use of information technology. The most distinct differences between an enterprise system and other transaction-oriented systems are that the enterprise system is a package versus a system custom developed in-house (implying long-term dependence on a vendor) and that embedded in the enterprise system are normative

business practices (requiring many adopting organizations to undertake some form of process reengineering). To date, collective experience with enterprise systems remains quite poorly codified; many organizations approach the phenomenon with little directly applicable knowledge and skill.

Whether or not enterprise systems will remain an enduring part of the organizational IT landscape clearly remains to be seen, but, because they have become such a large part of organizational IT infrastructure, they will continue to be a consequential phenomenon for some years to come. Enterprise systems affect nearly all aspects of organizational life, not only at the point of startup but also throughout their operational lives. Indeed, an organization's enterprise system affects its need and ability to upgrade or convert to more modern technologies. Consequently, we need a framework for understanding and analyzing these systems throughout an experience cycle that includes initial decision making, "development" and implementation, early use, and extended use. The framework outlined in this chapter is a first attempt at an integrated framework for understanding the systems intended to integrate organizations.

The key features of this framework include the following. First, it addresses both the motivated behavior of organizational actors, that is, the goals they are trying to achieve, and the factors outside their control, such as the performance of vendors and the reactions of customers and competitors. Second, the framework allows for both emergence—outcomes that are not deterministic but are influenced by both chance events and human actions—and dynamics—responses to problems and opportunities created by earlier decisions and actions. Third, the framework emphasizes the long-term nature of the enterprise system experience, including maintenance and future upgrades and conversions as major contributors to total costs and benefits. Fourth, the framework understands success as a multidimensional and relative concept and introduces the concept of optimal success to accommodate unintended consequences and external realities that are not fully represented in organizational goals. Fifth, as a process theory, the framework helps explain why organizations do not always achieve optimal success. Finally, the framework uses the concept of unresolved risk or variance to explain how errors can have consequences that show up long after the errors originally occurred. This explains why organizations often find it so hard to correct problems and to learn from their experiences with enterprise systems.

ENDNOTES

1. By the mid-1990s, it was common wisdom that business process reengineering in many companies had failed because of the difficulty and huge expense of reprogramming their core transaction processing systems to support the new processes.
2. See the chapter by George, this volume.
3. In the first draft of this chapter, we wrote: "The ERP total software licensing business is expected to grow to $20 billion by 2002 (Stein, 1998). And consulting firms have estimated that ERP consulting services will grow to $8 billion by the year 2002 (Stein, 1998)." David Brown, of Key Performance International, provided these interesting observations: "I noticed AMR Research released a new

report last week (5/18/99) projecting total ERP software revenues of $66.6 billion by 2003 (Enterprise Resource Planning Software Report, 1998–2003). I thought you might be interested in the large discrepancy between Stein's projections and AMR's. . . . Based on earlier studies by AMR, ERP consulting services range between 2 and 5 times ERP license revenues. AMR reported average ERP license revenues of $4,000 per seat (Baan or SAP) and ERP consulting fees of $8,000 (Baan) to $20,000 (SAP) per ERP seat. If AMR's report of ERP consulting fees is accurate, Stein's estimate of $8 billion in ERP consulting revenues supporting $20 billion in ERP license revenues is too low by a factor of 5 to 10" (Brown, 1999).

4. See the chapter by Weill and Broadbent, this volume.
5. See the chapter by Fichman, this volume.
6. The configuration of enterprise systems is not to be confused with (business) product configuration, a functionality supported by some ERP software extensions.
7. Davenport (1998) quite rightly points out that such lack of integration may be perfectly appropriate for some organizational structures and business models.
8. See the chapter by George, this volume.
9. In the language of SAP, installation work is divided into "applications" work (the province of end users) and "basis" work (the province of technical specialists).
10. For an example of some of the trade-offs involved and how they are resolved, see the case study of Microsoft's implementation of SAP R/3 financials (Bashein, Markus, & Finley, 1997).
11. Conversion may be unavoidable when the vendor redevelops the software for an entirely new computing architecture (e.g., mainframe to client/server, client/server to object oriented).
12. An industry association dedicated to issues related to production and inventory control.
13. When pushed, most analysts admit that there is no way to verify that their practices are really "best." Presumably, some organizations find ways to perform the process even better.
14. Conversely, the unwillingness to change how they do business requires many organizations to modify enterprise software.
15. See the chapter by Grover and Kettinger, this volume.
16. The difficulty stems in part from the lack of the relevant knowledge and skills, which are scarce and distributed across a variety of occupational specialties. Therefore, system integration often involves managing a sizable collection of technical experts from different fields and often from different organizations. New enterprise system developments such as Web-enabled software and outsourced implementation and operations are possible solutions to these problems.
17. To facilitate this integration, enterprise system vendors are now publishing the structure of their software, for example, SAP's business application programming interface (BAPI).
18. Most enterprise systems trace their roots to MRP II—a disciplined approach to managing discrete parts production. Many businesses differ substantially in form, structure, and business processes from discrete parts manufacturers, for example, process industries, service business such as repair operations, and manufacturers of products with "dimensionality" (e.g., clothing, footwear, telecommunications cables). (An inventory consisting of two 500 meter cables is *not* equivalent to one 1,000 meter cable.) Therefore, companies in particular

industry segments must buy a specialized version of the enterprise software created by either the vendor or by a third party. Otherwise, they face the need to modify the software at great cost and risk.

19. In some cases, these extensions are developed by the vendors. In other cases, the vendors have acquired third-party software firms to gain access to their products.

20. See the chapter by Fichman, this volume.

21. See the chapter by Fichman, this volume.

22. Even when enterprise systems are not adopted at a corporate level, they may be adopted by a corporation's individual business units.

23. See the chapter by Barua and Mukhopadhyay, this volume.

24. See the chapter by Kirsch, this volume.

25. See the chapter by Grover and Kettinger, this volume.

26. See the chapter by Olfman, this volume.

27. See the chapter by Kirsch, this volume.

28. See the chapter by Ang and Slaughter, this volume.

29. See the chapter by Fichman, this volume.

30. See the chapter by Robey and Boudreau, this volume.

31. In other words, our level of analysis is different from Alter's (1999).

32. See the chapter by Barua and Mukhopadhyay, this volume.

33. An argument in favor of the contested definition is that many enterprise systems projects are scaled back during installation, which usually means a failure to achieve the ambitious original goals. On the other hand, this definition would also count as failures projects that yielded unanticipated benefits.

34. This system and concomitant organizational changes succeeded in reducing accounting personnel by 70%, but morale plummeted, skilled people left the company, and the system was disinstalled when the company was acquired.

35. These metrics include business, human, and adaptability outcomes (Silver, Markus, & Beath, 1995).

36. Hirt (Hirt & Swanson, 1999) studied an organization in a declining industry with severe profit pressure that implemented SAP to reduce mainframe computer costs and solve the Y2K problem. Because the company had done some process redesign before implementing SAP, it did not do more reengineering when adopting SAP. Later, though, the company wondered whether more reengineering would have produced better gains.

37. See the chapter by Agarwal, this volume.

38. See the chapter by Robey and Boudreau, this volume.

39. See the chapters by Barua and Mukhopadhyay and by Robey and Boudreau, this volume.

40. This phase model differs from that proposed by Ross (1999).

41. We borrowed this term from Nidumolu (1995).

42. Perrow (1972) would argue that tight coupling increases the likelihood of various kinds of failures.