1

WHY MOST IMPLEMENTATIONS SHOULD BE RAPID BUT ARE NOT

There is a tremendous need for a different approach to implementing business computer applications. Instead of large, full-scope projects, organizations should be doing a lot of smaller, focused projects. Historically it has taken many months, and even years, to complete these implementations. In the future these projects will have to be done in a matter of weeks and months.

There are important business reasons why the old way of doing these projects will have to change. Organizations are now in a business environment where their success will depend on their ability to rapidly respond to changing business requirements. There are no alternatives to becoming better at making quick changes. Business changes require changes in business processes and the systems that support them. Therefore, organizations must learn how to rapidly implement new business application systems.

This chapter lays the groundwork for the material that follows. We begin by making a case for rapid implementations. Then, we establish a foundation for the types of applications that are being implemented by looking at the journey that standard software has taken over the last 40 years and describe an architectural framework for looking at the new applications. The chapter then examines the difference between installing and implementing software. After that, we look at how fast is rapid. The chapter closes by laying out the key success factors for rapid implementations.

NEED FOR SPEED

The way that organizations have traditionally implemented complex business applications has to change. The old approaches are too slow, expensive, and unreliable. In an age of rapid, continuous change brought about by the Internet, reengineering, and globalization, projects that take years before an organization starts receiving benefits are no longer acceptable! Recent changes in business and technology have permanently raised the implementation bar.

As a result, organizations need to find ways to speed up the implementation process to solve their business problems and take advantage of new opportunities. With the availability of new tools and resources for these projects, taking a rapid approach to the implementation of standard business software is possible. For most situations, it is the preferred approach.

Business applications (e.g., accounts payable, order entry, manufacturing, payroll, shipping, inventory management, loan processing, student registration, customer billing) support, and to a certain extent determine, the way work gets done in organizations. In the past, midsize or larger organizations typically took one to three years to implement suites of these applications. In most cases the organization did not receive any benefits until the entire project had been completed and the organization went live with new software and business processes. In the new economy, organizations must figure out how to solve problems and get benefits from these systems in a matter of weeks or months.

To achieve this goal requires a different way of looking at these projects. The desired results should drive the solution. Instead of starting from how long these projects have taken in the past, we must let business requirements determine how long they *can* take in the future. The key question should be: How long can the organization take to change business processes and the systems that support them in order to meet the increasing demands of the marketplace?

Once they know the target, organizations must get serious about figuring out how to get these implementations done quicker—in the required time frames. Increasingly, parts of complex systems will need to be implemented in two to six months in order to meet competitive pressures. Fortunately, there are already many examples that prove that, with a different approach and new tools, meeting these shorter time frames is possible.

The drivers for business change (and faster implementations) are everywhere. There are tremendous changes occurring in the business (and nonbusiness) world as a result of new technologies, business models, global interests, changing customer expectations, and competitive challenges.

As evidence of the magnitude of these changes, you have only to consider the number of new e-businesses that have been created in the last three years. These companies cannot wait years to get financial, operational, and human resource systems up and running. They need to quickly get systems operational to take orders, ship goods, pay employees, and produce financial statements for investors.

In addition, existing organizations are all scrambling to become e-businesses. New technology enablers are forcing most of them to reevaluate their business models and relationships, change organizational structures, and redesign business processes to be more responsive to the increasing demands of customers. Internet technologies are creating many new possibilities.

No company can ignore the availability of new technologies that, perhaps for the first time, truly support the redesign and implementation of new ways of doing business. The only question is how they can take advantage of these new technologies. Most organizations have not been very successful in doing this in the past; many have recently gone through painful implementations of new systems.

The good news is that there are now software packages (standard software) for enterprisewide and external-facing applications that leverage the newer technologies. They support business processes that we would not have dreamed of five years ago. With these packages, organizations will not have to develop their computer applications from scratch in order to create some tremendous capabilities.

The bad news is that we are going to have to implement these new technologies in much shorter timeframes than ever before. Organizations must figure out how to do successful rapid implementations of new processes using these new packaged solutions.

CUSTOM DEVELOPMENT (1960s AND 1970s)

Before we go any further in our discussion of rapid implementation, we need to set the stage by briefly examining how organizations have implemented these packages in the past. To do this right we need to briefly look at the history of ERP, what is meant by ERP and e-business applications, the reason for implementing them, and the distinction between installing and implementing standard applications. We begin in this section with the early use of computer applications.

When computers first started to be widely used in a business setting, in the early 1960s, they were used to automate simple, routine business tasks. Payroll was a natural place to start. The calculation of payroll amounts (multiply hours worked times hourly rate and then subtract for taxes and deductions) was not particularly complicated but had to be done many times every week or so—and very accurately. The next areas to be automated were the financial applications: general ledger, accounts payable, accounts receivable, and fixed assets. Because of these historical roots, many of the IS departments in organizations historically reported to the controller or vice president of finance.

The computer excels at doing things very fast and doing them the same way every time. Those are two things that we humans, no matter how hard we try, generally have problems doing. (That is especially true when we are tired or do not feel particularly motivated.) Therefore, computer systems were widely used to automate back-office functions.

These early business applications had to be programmed by people from the IS department, or by external consultants. When a computer was purchased, it could not do anything of value for the organization until the IS department wrote some programs. So, every company wrote its own general ledger, payroll, order entry, and billing systems. Unfortunately, IS departments generally were not very successful in creating these systems. As a result, they often gained a reputation for delivering systems that did not meet users' needs, cost twice as much as planned, and were delivered late.

Developing business applications (or any other kind of computer program) is a complex task, one that most organizations did poorly. There were reasons for this situation.

The first programmers were usually people from one of the functional departments who had to learn programming on the job, with little training and supervision. It is a very complicated activity to determine business requirements, design and code programs, determine layouts for data records, and test all the things that should go right and could go wrong. These tasks are especially difficult when the programmer is under a lot of pressure to get the application out quickly, which was always the case because of the multiyear backlog of requests in the IS department.

Also, by today's standards, there were not a lot of good tools available to develop the first systems. The first programmers did not have high-level programming languages, relational database management systems, a lot of memory on the computer to work with, or easy ways for a program to interact with the user's terminal. There were no graphical interfaces back then; it was even difficult to program real-time dialogs with terminals using character-based interfaces.

To make matters worse, after only short discussions with several of the users and managers, the programmers often disappeared into their cubicles to develop these new systems. This development was often done at night when programmers could get time on the computer to compile their programs. So, the development was done without a lot of further involvement from the people who would ultimately use the system or its reports.

As a result, the requirements for the applications were often misunderstood by the programmer (who probably did not have a business background and was guessing what the user wanted), the system did not meet users' needs, and it was delivered late and over budget. Some things have not changed a lot over the years.

STANDARD SOFTWARE (1970s AND 1980s)

Around the 1970s, some of the programmers and computer consultants started to wonder if there was not a better way. Why was every company reinventing the wheel when it created business applications? After all, is the general ledger or payroll application of one company that much different from one needed by other companies? So, companies were started to create *standard* (or *packaged*) software. This is software, perhaps developed for a particular company initially, that is then sold to other companies and adapted for their use.

Using this type of software usually shortened the time for developing and implementing new business applications; after all, the software had already been programmed, tested, and debugged at another company. In addition, using this software leveraged the talents of application designers, experienced programmers, database and hardware gurus, training course developers, and others that most companies could not afford to hire or retain.

For example, five IBM employees in Germany left to start a company to create standard financial software in 1972. That company, SAP, grew to have revenues from software sales and services of over $5 billion in 1999. Other companies were started to create standard payroll and human resource (HR) applications, or any number of other business applications where it seemed possible that a standard set of software could be used by a number of organizations. Even some of the computer hardware vendors wrote standard applications that they could sell to their customers.

These software packages were licensed and implemented by the organizations, usually after they had been *modified* (the programs from the vendor were changed) to meet the unique needs of the organization. Unfortunately, these modifications meant that the software vendor could no longer support the package. There was no way for the vendor to know what changes had been made by the customer. This also meant that the IS organization usually had to reapply these modifications each time they implemented a new release of the vendor's system. The only time this was not necessary was when the new release included new functions to handle what had been supported by the modifications.

In spite of the difficulty this caused when applying new releases of the vendor's package, buying and modifying software was still quicker than starting from scratch. So, the organization assumed responsibility for maintaining the applications after they were implemented.

These standard software vendors started out by producing and supporting a limited number of application modules. Some companies sold only financial applications; others concentrated only on manufacturing modules. In its area of focus, each vendor competed to be considered the *best-of-breed.* For example, in the 1980s many people considered Management Science of America (MSA) and McCormick & Dodge as the two best-of-breed financial package vendors.

As a result of this specialization, if you wanted to get a complete set of applications for your organization, you might buy a general ledger module from one vendor, a payroll module from another, and a loan-processing module from a third. You might also have your internal IS staff design, code, and test four other modules that you needed, but could not find a suitable package for on the market. Most of these modules ran on mainframe computers, operated in a batch mode, and were extensively modified (because of missing functions the organization had to have to do business). Each had its own separate database and there were no real-time interfaces between the packages.

Any interfaces for sharing information between the applications had to be programmed by your own IS personnel or consultants you hired to help you implement these applications. When you implemented a new release of the vendor's package, the interfaces often had to be changed for all the other applications the updated one touched.

APPLICATION SUITES (1990s)

Over the years, the vendors added more and more modules to their suites of applications—acquiring some from other vendors or developing new modules in conjunction with customers. In addition, the applications started to get better functionally as the vendors invested large sums of money in their development and responded to current and prospective customers' demands for new functionality in the next release. The standard applications started to represent best practices in how to use automation to support business processes, as the vendors took the best ideas from all their customers and started to incorporate them into the products.

The vendors also started to change the technology architectures for the applications. They started developing their applications so that they would run on a number of hardware platforms, operating systems, and database management systems. They also started to integrate their various applications so they could use common databases and share information in real time—as the transactions occurred.

By the late 1980s, vendors started to offer fully integrated suites of applications that supported many, or most (depending on size, complexity, and industry) of an enterprise's functions. This created some basic choices. An organization could choose to buy modules from different vendors to get best-of-breed applications that they would have to integrate themselves. The other option was to buy a suite of applications from one vendor that was integrated out of the box.

Because of the advantages of integrated applications and the lower training and maintenance costs associated with going with one vendor, many companies started to go the integrated suite route. (This is the same reason that many organizations use Microsoft Office instead of discrete best-of-breed productivity tools from several different vendors.) What we really had by this time were products from a single vendor that supported all the planning, operational, and resource management functions of the entire enterprise. They were not called ERP systems until later, but the vendors were well along the way to realizing that concept.

In the 1990s ERP became hot. Integrated applications were so desirable and the packages had such extensive functionality and technical capabilities that companies started to replace their homegrown legacy systems with ERP packages. The bigger companies, especially manufacturers, started doing this first and ran the new systems on their mainframes or minicomputers (e.g., HP3000, DEC Vax, or IBM AS/400).

However, the buyer still had to be careful to separate hype from fact. Vendors sometimes announced modules as being available before they had completed developing and testing them. In addition, some of the packages did not include all the functions an organization wanted or needed to do business in its industry. Also, all the package vendors started calling themselves ERP vendors, even if they had big holes in the scope of modules they offered (i.e., no payroll/HR modules or no manufacturing modules).

Throughout the 1990s the major ERP vendors—Oracle, SAP, J. D. Edwards, PeopleSoft, Baan, Lawson, and QAD—had tremendous growth. The demand for these products was driven by the high cost of maintaining legacy systems, the desire for new functionality that could not be provided by in-house developers, the need to handle Y2K issues, and the push of globalization and competition. The three years before the end of the century were great years for ERP vendors. Most of the top ERP vendors had initial public offerings (IPOs) in the late 1990s and the owners and employees did very well. The major players were growing at 40 to 60 percent annual rates. However, storm clouds were forming.

ERP AND E-BUSINESS (2000s)

The months leading up to January 2000 produced shock waves throughout the application software industry. By that time, it was too late for their customers to begin implementing new applications to take care of Y2K issues (many companies had done this earlier rather than fix old legacy systems), so customers put the purchase of new ERP modules on hold until they got through the first months of 2000. Then, when the capital funds were released that spring (and Y2K turned out to be a nonevent), the money was allocated instead, at least in the U.S. market, to a new phenomenon that had captured, by storm, the imagination of business leaders and the general public: e-business.

To be fair, the ERP vendors had been preparing for aspects of e-business for several years. By 1999, most had Internet-enabled their applications (i.e., you could use a browser on the Internet to access information in the ERP system and even perform some transactions). And these vendors were rewriting their systems to take advantage of Internet technologies to deliver new functionality powered by the Internet.

In fact, by this time it was difficult to find one of these vendors still calling themselves an ERP vendor. Almost overnight, they all started to call themselves *e-business vendors.* However, most of the attention in the e-business software space was being given to non-ERP and Internet pure-play vendors. Companies like Siebel, i2, Ariba, BroadVision, and Commerce One had best-of-breed, point solutions that were powering e-business and other advanced applications that everyone was excited about.

Companies were spending hundreds of thousands of dollars on this new software and much more on consulting assistance to implement these applications and the technical infrastructures on which they ran. Gartner Group, a research organization, put the average price of putting up a complete commercial web site for a large or mid-size organization at $1 million. And this is before organizations added the complex e-business applications.

However, there was a lot of smoke-and-mirrors associated with the use of these new applications. It was not uncommon for a company to have a great e-store site powered by one of these new packages with shopping carts, credit card authorization, and quick checkout functionality. But, behind the scenes, the orders taken on this web site were printed out and either faxed to a distributor or rekeyed into the organization's ERP order entry screen to handle financial and logistics processing.

Most of these point applications were not integrated with the backbone transaction systems of the organization. In the case of the dot-coms, there often were no transactional systems in the background. This situation makes it very difficult for customers to check on the status of their orders from the web site.

Companies had three basic options for their e-business and advanced applications:

1. They could develop them in-house (programming in HTML, C++, or Java) and interface them to the existing ERP or legacy systems.
2. They could buy best-of-breed products with great functionality and then interface them to existing systems.
3. They could wait until the ERP vendors came out with new e-business modules.

A lot of companies started down each of these paths.

Those who started down the third path did not have to wait long. In 2000 most of the major ERP vendors introduced e-business suites of applications that would significantly expand the functionality of their ERP systems. For example, Oracle released its new applications in May, called version 11i, as part of what it calls its *e-business suite.* This suite of products has the new versions of the traditional ERP modules along with new modules for customer relationship management (CRM), supply chain management (SCM), and other functions that are truly state-of-the-art e-business: iStore, iSupport, Exchange, auctions. They even introduced remote access to Palm devices using Internet messaging for entering and receiving ERP information. In addition, all the functions of 11i have been written for and run on the Internet.

These are pretty impressive applications. If you want to see how iStore works go to the Oracle web site and buy something on their store: they use all their own software internally. Or look at mySAP.com. These new applications may not have all the functions and bells and whistles of the best-of-breed products but they have more than enough for many organizations. And they have the added advantage of coming, out of the box, fully integrated with the other ERP modules from the vendor.

Modules in these suites use the same databases and have real-time sharing of information between applications. And if you upgrade to the next version of these modules, you do not have to spend a lot of time and effort reimplementing the interfaces between all the components they touch. Studies show that companies who go with best-of-breed solutions spend a significant portion of their IT budgets maintaining all the versions of these components and the interfaces between these components.

However, it cannot be ignored that the best-of-breed packages like Siebel, Ariba, and i2 are great products. They have functionality that the full-suite e-business vendors will have difficulty duplicating. And, if this additional functionality is important to an organization, they can still be the right solution—even given the headaches of the interfaces. Even this aspect will get better as all the vendors move toward more standard interfacing schemes (using Internet tools like eXtensible Markup Language or XML) and the vendors and third-party software developers build standard interfaces between the major packages. A whole new generation of tools called enterprise application integration (EAI) middleware is being developed to ease the integration issues.

It is clear that the boundary between what we have traditionally referred to as ERP and the new advanced-function packages and e-business applications is blurring. Traditionally, ERP was used to designate the collection of fully integrated modules that provided automated support for most of the functions of an enterprise. But ERP vendors never supported all the business areas well. Most did not have good functionality to support a number of areas needed by organizations; sales force automation, call centers, advanced planning and scheduling, data warehousing, and analytical analysis are a few examples of poorly supported functionality.

The primary focus of ERP vendors initially was on the needs of manufacturers and distributors. Even the term ERP is a successor to the manufacturing term MRP (materials requirements planning) and came into use when vendors started to fill out the suite of applications provided beyond the manufacturing areas. Throughout the 1990s, ERP vendors provided modules in four primary areas: (1) financial, (2) distribution, (3) manufacturing, and (4) human resources. But this left many major functions in other industries that are just starting to be addressed by these vendors with vertical market and industry solutions.

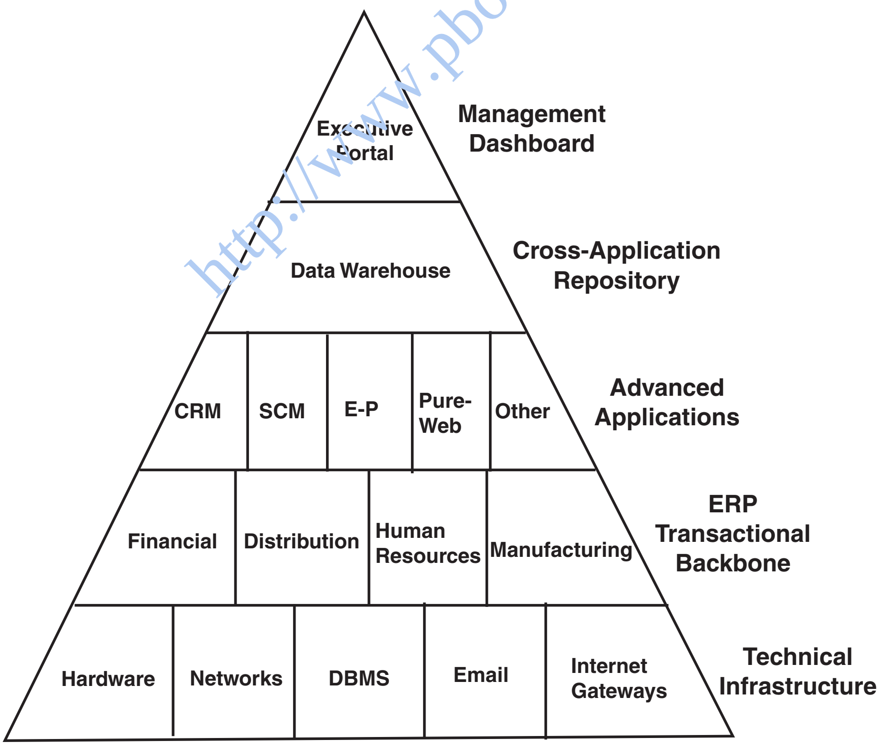
In addition, the Internet has enabled new functions like e-procurement and e-retailing that were never practical on a widespread basis before. Companies have done electronic data interchange (EDI) and electronic funds transfer (EFT) for years. But the use of these technologies was limited to a small number of large organizations and their suppliers. These technologies were too expensive for smaller organizations.

A key business requirement is that all of these applications, old and new, be integrated. Integration requires that the different applications talk to each other in real time—not in periodic batch updates. Also, each of the applications should not have separate databases. All of the data needed for all the systems should be together in one database so that there is no redundant data and only one definition of what each data item means. All of this should be done seamlessly: the modules should have the same look and feel, users should be able to get to any of the information in the system in real time (as soon as a transaction has occurred), and the IS department should not have to do a lot of work to make the integration happen.

APPLICATION FRAMEWORK

As you have noticed, there are significant terminology problems when business application modules are discussed. Applications have already been referred to as ERP, e-business, CRM, e-stores, e-procurement, SCM, and data warehousing. Overlaps in the traditional boundaries associated with these terms have also been discussed. Figure 1-1 is an attempt to show how all these different types of applications fit together. It provides a common framework for use throughout this book.

FIGURE 1-1 eXtended Enterprise System Framework



I will use the term *eXtended Enterprise System (XES)* to represent all the applications shown in the figure. Other people have proposed terms like enterprise applications (EA) and enterprise systems (ES) to be used in place of the outdated ERP acronym. But neither of these terms gives proper credit to the role that these systems play in extending business information and processing beyond the four walls of an organization. Enterprise resource planning supported the enterprise, and a lot of organizations still need to successfully implement ERP functionality. eXtended enterprise system include the ERP functions and all the other applications that extend the organization's systems throughout the supply chain—to customers and suppliers. XES represents all the enterprisewide and cross-enterprise applications that have been or soon will be e-enabled as organizations execute their e-business strategies.

TECHNOLOGY INFRASTRUCTURE

Figure 1-1 shows the relationship between the components of a business application architecture and clarifies where the ERP and E-business systems fit into the overall structure. The bottom layer of the architecture is the technology infrastructure that must be in place to support all the other applications. This infrastructure can be developed and maintained in-house or can be outsourced to a number of different companies such as application service providers (ASPs), telecommunication providers, and consulting organizations.

The hardware component refers to the parts of the infrastructure you can touch. It includes, first of all, the computers (mainframes, minicomputers, workstations) installed to function as servers (e.g., database servers, application servers, print servers, mail servers). However, it also includes PCs, workstations, handheld devices, data storage devices, printers, scanners, bar code readers, radio frequency transceivers, and various communication devices (routers, bridges, gateways, and modems).

The networks in the technical infrastructure are of two types. The local area networks (LANs) connect PCs, printers, and servers within a single location. The wide area networks (WANs) connect all the remote locations and LANs together into one corporate network. The WAN includes communication links between the various locations (e.g., public telephone, dedicated telephone services, Internet, virtual proprietary networks, and private networks) that use a combination of copper lines, fiber, microwave, or coaxial cable links. It also includes the network operating systems used to control the hardware and application software.

The operating systems of choice for XES systems are either UNIX (with various proprietary flavors from each of the hardware vendors) or Microsoft NT. Some of the XES vendors have recently announced support for the Linux variant of UNIX; Linux is getting a lot of interest and support, mainly from those who believe that the openness of UNIX has been tainted by hardware vendors who have added their own proprietary features. Other supporters of Linux just want to have a viable alternative to the Microsoft family of products.

The database management system (DBMS) is the software that controls access to the data stored and used by all the applications. For XES systems this database software is usually a relational database system from either Oracle (8i), Microsoft (SQL Server), or IBM (DB2). People called database administrators (DBAs) are responsible for maintaining and tuning the databases, which retain all the information needed to process the XES applications.

Email has become a critical component of the IT architecture. Its use goes far beyond the ability to communicate informally within the organization and with key external partners (i.e., customers, suppliers, regulatory organizations). Email is becoming the means by which workflow functionality in XES applications notifies individuals when they should take action. This transaction approval and exception notification functionality is part of the normal process flow in many of the vendors' applications. It is also used to alert managers of performance metrics that are outside normal tolerance levels.

Email is also becoming an important tool in marketing to customers and providing services. Systems are now available to assist larger organizations in routing the thousands or hundreds of thousands of inbound (and outbound) messages each day to the appropriate individual or department to facilitate the prompt and efficient response to these messages.

The final component of the technical infrastructure layer is the hardware and software used to provide access to the Internet for employees as well as to make it easier for an organization to share information with its customers and suppliers. In the future, this area will also need to address the way that messages and transactional data can be communicated to remote devices such as laptops, Palms and other personal digital assistants (PDA), and other handheld devices using cellular and Internet capabilities.

APPLICATION LAYERS

There are two application layers in the XES framework—the ERP transactional backbone and advanced applications. In the past, different vendors have provided these types of products. Today, the traditional ERP vendors are providing both categories of applications. In addition, the advanced application vendors are partnering with other vendors to provide wider coverage for their products and easier ways to integrate them with other applications.

The ERP transactional backbone layer includes the application modules that have been offered by ERP vendors for the last 20 years. In the financial area the vendors have offered modules for general ledger, accounts payable, accounts receivable, fixed assets, cash and treasury management, and financial consolidations. These modules are available as standalone applications or as part of an integrated suite. The financial products from several of the vendors are good enough to be considered best-of-breed applications.

The distribution modules include order entry, purchasing, inventory management, and shipping. In many cases these products provide multisite functionality to support distribution requirements planning (DRP). With DRP, organizations can check on inventory availability and fill orders from multiple locations. They can also track and account for the transfer of materials between sites.

The order-entry modules are the ones most likely to be modified by an organization during its implementation. Organizations use a wide variety of methods to price their orders. No matter how many options are provided in the standard software, organizations seem to come up with industry practices that are not supported. Therefore, the vendor's code is sometimes modified, or custom code is interfaced to the standard application, to handle requirements in this area.

The distribution area is also one of the first areas to be Internet-enabled. Customers want to be able to check on the status of their orders. Giving them browser access to this information over the Internet was a development priority for most of the vendors.

The human resources area includes personnel, timekeeping, payroll, and benefits modules. Each vendor combines these modules into categories a little differently. Also, once you get away from the top-tier vendors, there are sometimes significant gaps in the completeness of the suite of modules offered by a particular vendor.

One of the topics that will be discussed in Chapter 8 in the hosted applications section is the option of outsourcing the processing of applications. This is not a new option. For many years, small and midsize organizations have outsourced the processing of their payroll and human resource applications. The outsourcing trend is just extending the scope of applications that are hosted by external organizations.

The manufacturing modules include plant scheduling, capacity requirements planning, engineering, bill-of-materials and routing maintenance, dispatching and shop floor reporting, material requirements planning (MRP), and standard costing/variance reporting. These modules were developed to support a number of different types of manufacturers. The requirements to support a company manufacturing chemicals is different from those necessary to support a company assembling airplanes. Therefore, these are often very complex modules to develop and test.

Several of the packaged software vendors called themselves ERP vendors before they had manufacturing modules. They addressed this inconsistency by acquiring the manufacturing modules of other vendors. However, it took several years before these modules were integrated with the original modules in the vendor's suite.

The next layer in the framework includes applications that provide advanced capabilities in specific functional areas. Many of these applications include capabilities that are made better, or perhaps only made possible, by extensive use of the capabilities of the Internet. These modules are the primary arenas for competition between best-of-breed package vendors and full-suite XES vendors.

These advanced applications have been available from best-of-breed vendors for several years. Many of the best-of-breed companies were formed in the 1990s. The products from these vendors have functionality that is not available in the offerings of integrated-suite vendors. In the past, many of these modules were not even available from the traditional ERP vendors. However, that situation is rapidly changing.

Oracle, SAP, J. D. Edwards, and several other major vendors are now offering modules in these advanced function areas as part of a fully integrated suite of e-business applications. Some of the vendors have developed their own modules from scratch; others have acquired the products of niche vendors. There are significant differences with regard to the capabilities of these modules and the level of integration between them and the other modules provided by the vendors. These advanced application modules fall into the areas of customer relationship management, supply chain management, e-procurement, pure-web applications, and a catch-all called other.

Customer relationship management is a particularly hot area of interest from both the business strategy and technology perspectives. Customer relationship management is about supporting all the customer-facing processes in the organization. Its goals are to increase sales and customer loyalty and retention, and to enhance customer service.

The four major types of CRM modules are: (1) sales force automation, (2) marketing, (3) call center management, (4) customer service and support. There are best-of-breed products in all the major areas associated with CRM (e.g., marketing (e.piphany), sales (Siebel, BroadVision), and service (Clarify, Silknet)). However, there are also integrated solutions from the XES vendors that address these same areas and use the Internet to do so. For example, Oracle has (as part of its 11i version) CRM modules such as Marketing Online, iStore, iSupport, Contracts, and E-Mail Center.

Supply chain management includes the processes to manage the relationships among all the entities that are involved in sourcing, manufacturing, and shipping products to the end consumer. These relationships include raw materials and component suppliers, manufacturers, distributors, warehouses, transporters, retailers, and the ultimate consumer. The goals of SCM are to share information in order to drive unnecessary inventory out of the supply chain and to enhance the effectiveness of the supply chain partners in being responsive to the needs of the marketplace.

To achieve these goals, enterprises strive to coordinate the planning between the various supply chain partners and enable collaboration between the various entities. Obviously, this involves a lot more than technology enhancements and giving suppliers access to an organization's master production schedule and orders. There are major organizational, cultural, and policy issues involved with these initiatives.

The major types of applications in SCM are advanced planning and scheduling (APS), warehouse management, and transportation. In APS, the two major best-of-breed packages are from i2 Technologies and Manugistics. However, SAP, J. D. Edwards, Oracle, and several other XES vendors have added APS modules to their product lines.

The goal of e-procurement (e-p) is to use Internet technologies to make the procurement process more efficient and less costly to the participants. There is a lot of room for improvement in this area. It can cost hundreds of dollars to process each requisition and purchase order.

The buyers (and people in accounts payable, receiving, and other areas of the organization) spend a great deal of time tracking down and solving problems with individual transactions (e.g., determining why a supplier has no record of an order, expediting late receipts, investigating unauthorized purchases from unauthorized vendors, reconciling invoicing differences). Often they do not have the time to do the sourcing, vendor analysis, and contract negotiation activities that would really add value to the organization.

The Internet helps in this area by supporting a self-service approach to purchasing. Through web catalogs, exchanges, and workflow technologies much of the requisition, approval, ordering, and payment process can be automated. One of the best-of-breed solutions in the area of nonproduction-item purchases (known as maintenance, repair, and operations (MRO) items) is from Ariba. In addition, a number of Internet-powered exchanges are being set up to handle other types of material purchases.

Auctioning functionality is being extended beyond consumer purchases and is becoming popular in the business-to-business (B2B) purchasing space. The XES vendors are offering this same functionality. These applications sometimes have the same look and feel of the B2C (business-to-consumer) offerings (i.e., some of the vendors are using the shopping cart analogy and checkout process for B2B purchases).

Pure-web includes applications that are only practical because of the maturing of the Internet and its associated technologies. Although it could be argued that some of the things that have already been discussed could fit into this category, what are included are things like e-retailing and Internet remote messaging, applications we never thought were possible, even a few years ago.

Amazon sells an amazing number of books worldwide and records every click made on their site for future use in personalizing the buying experience. Dell allows us to configure our own PCs online and then builds them one at a time using functionality in its backbone systems. Cisco has effectively automated its sales and service functions through self-service functionality on its web site. All these e-retail activities are possible through the use of point solutions like BroadVision—or from the new modules of the XES suite vendors.

What is truly amazing is the potential for remote processing through the use of Internet messaging. At one of the XES vendors' demonstrations, a customer (at the customer site) asked the salesperson whether 20 items could be shipped next Tuesday. To answer this question the salesperson would normally have to telephone someone back at the plant or go back to the office and check things out on the computer. Instead, the salesperson went through a series of screens and data entry fields on a Palm PDA, transmitted the order remotely through the Internet to the XES applications back at headquarters, and received a confirmation, again on the Palm, that the order could be shipped—all in less than a minute.

The final category of applications at the advanced application layer is called Other. It includes a number of different things, including industry-specific applications that are not currently available from the XES vendors.

Each industry has its own unique requirements that often are not adequately addressed by the standard suites of software. Examples include a loan processing system for a bank, a student registration system for a university, a laboratory system in a hospital, and a mutual fund tracking system in an investment company. In the past, many of these systems were developed from scratch in-house by the IS department, often with the assistance of consultants. In some cases these systems were purchased from niche software companies that focused on developing software to meet the unique needs of a particular industry or industry segment. Many of the XES vendors are now beginning to develop these industry-specific modules.

In this category are also the systems that were custom developed by organizations in the past to meet unique needs or provide competitive advantages. Some of these legacy systems may not be replaced by modules in the standard software and, therefore, must be interfaced with the other applications. In certain cases these applications offer real competitive advantages. These advantages might be lost if a company were forced to use the generic functionality of the standard systems in particular areas.

The final type of applications in this Other category are the knowledge management systems that are becoming increasingly important to a wide variety of organizations. These are the repositories of intellectual capital—the knowledge and best practices of the organization. They allow the organization to share knowledge across functions and geographies. This knowledge is maintained in groupware and collaboration products like Lotus Notes, Microsoft Exchange, and other web-based repositories.

INFORMATION PORTALS

At the top two layers of the framework (i.e., cross-application repository and management dashboard) are those components that transform information from the various subsystems into a form for easier analysis and review. Increasingly, this information is being made available over the web in the form of a browser-based portal. There may also be separate portals for the CEO, CFO, and other groups of managers in the organization. Each management role requires different types of information to make decisions and track performance. A portal makes all the various information views available at any time, from any location.

At the layer above the applications is the *data warehouse.* There is a lot of valuable information in the various databases of the ERP and advanced applications, in customer responses to an organization's web site, and in various external databases. This information is at the detailed, transaction level and these databases are optimized to support transaction processing. Because of the potential negative impact on response times the organization would not want to do a lot of ad hoc queries and reporting against combinations of these databases.

Many organizations have found value in extracting information from the various application databases, at either a detailed or summary level, and putting this information into a separate database in a structure and format that facilitates ad-hoc queries, reporting, and analysis. This special database is called a data warehouse and there are special tools from vendors like Hyperion that can be used to analyze this data. The major ERP and E-business vendors also provide data warehousing applications with automated interfaces to their modules to make the extraction easier.

At the top layer of the XES framework is the *executive portal.* This is the radar screen or dashboard for top management. It tracks the key performance metrics of the organization, at a high level, and keeps the pulse on how the organization is doing overall. It also identifies situations that should be brought to the attention of the top managers of the organization. In many cases these applications implement a balanced-scorecard approach to management reporting.

These portal applications are tailored to the unique needs and preferences of the managers who will use them. The output of these applications is often graphical (i.e., pie charts and bar charts and colors to highlight items that are out of tolerance). These top management applications are becoming increasingly valuable in spotting trends and problems early, so they can be addressed in the rapid timeframes required by the new e-business economy.

Now that we have a framework that defines the type of applications that are candidates for rapid implementation, we need to draw some boundaries around these projects. The next section addresses the misconception that the only way to implement these applications in short timeframes is to ignore some of the key tasks in the implementations.

INSTALLATION VERSUS IMPLEMENTATION

Any discussion of rapid implementations must establish an important distinction. That distinction is the difference between merely *installing* software versus *implementing* a new system to support redesigned business processes. When rapid development is described in this book, it is in the context of the implementation option.

However, there are advantages in spending a little time describing the installation option.

An organization can put in new standard software applications very quickly if they are willing to take a number of shortcuts:

- Use a predefined configuration of the software without any modifications to the configuration parameters, source code, or reports.
- Transfer data directly from old systems without a great deal of data cleanup (or do no transfer of data).
- Leave out significant end user or IS personnel training before the new system goes live.
- Have people basically do their work tasks in the same way as they did them before, after going live with the new system.

This approach has often been used when there are excessive budget or time constraints on an implementation. However, this approach basically "paves the cow paths" with new technology while leaving business processes unchanged. It is surprising that any organization would take this approach. After all, if the organization does not change its business processes, how can it get significant business results from the installation?

There are certain situations where using the installation approach may be appropriate. For example, many organizations needed to install new applications rapidly during 1999 because their old systems were not Y2K compatible. In some cases these companies waited so long to start these projects that they had no option other than taking an installation strategy. They often did so with hopes that this approach would at least get them through the first few months of 2000. After that time, many of these organizations planned to go back and do a more complete job of implementing the new systems.

A second scenario could be a company running systems on old hardware and application software that are no longer supported by the vendor. These systems are often very expensive to maintain. In this situation, the company's goal may be to just get up and running on state-of-the-art hardware and applications—and address business process changes, data cleanup, and training at some future date.

A third example could be a startup organization that has no systems, no data, and no existing procedures. Often, the enterprise needs to get some applications in place immediately to handle some basic transactions processing. In this situation, the company must get something up quickly. The systems being implemented may not be strategic at this time and may not even get a lot of attention, support, and visibility in the organization.

The basic problem with each of these examples is that the organizations invested in new systems and technology but got no business return from their investments.

Additionally, these initial projects may have put future projects, and potential returns from these types of investments, in jeopardy.

The companies in all the examples probably are running on applications that people do not understand and with data that is incomplete and inaccurate. People, in this situation, will not trust the new system and will start to do their work with workarounds—using homegrown tools at the local level. As they work around the new system, they will stop feeding all the relevant information into the official system. There is not enough time to enter information into their spreadsheets *and* the new system—and people know which one they are using to support their work.

The organizations that use these short-cut approaches to put in new business applications never get around to doing the follow-on project to finish things like training and process redesign that were out of the scope of the initial efforts. The organizations in the first two examples may end up worse off than before the installation project began.

The end result in the last example depends on what happens after the initial project. A startup does need systems quickly and its people are used to learning fast and working incredible hours to do what has to be done. So, getting something up quickly matches the culture of the organization. And, if the new system does not work exactly as they want it to or does not do all the things that really need to be done, they can live with that. Often that is the least of their concerns. The major concerns are survival in the short term, attracting venture capital, and taking the company public.

However, as these startup organizations grow they soon get to the point where they need robust, full-function systems and need to look at the efficiency and effectiveness of their business processes. Then the challenge is to change the management style and culture around standardized, optimized business processes. Sometimes this transition requires an infusion of new people into the organization with experience from more stable environments.

If an organization goes to all the trouble and expense to purchase and implement a new system, it hopes to get some return from the effort. After all, there will be a lot of headaches and challenges as people are forced to learn a new system with different screens, transaction codes and terminology, and a new look and feel. So, why would any organization go to all this trouble? The answer is to get business benefits.

These business benefits come in a variety of forms. Some are related to cost reductions. Organizations often are looking for ways to reduce operational costs by streamlining business processes and cutting out non-value-adding steps, intermediaries, and jobs. They want to take inventory not only out of their organization, but from the entire supply chain. They want to be able to subcontract or outsource functions that are not core competencies. They want to move toward self-service functions for their customers, suppliers, and employees.

Other benefits are related to increasing customer service and responsiveness. Most organizations are interested in reducing cycle times: how long to take an order, process a claim, take an application for new service, schedule students for the fall term, or produce a patient bill. They want to make it easier and more enjoyable for customers to interact with the organization. And most organizations are interested in increasing sales through better understanding of their customers—what they want, which ones are profitable, how the organization can get more of their business.

To achieve these types of benefits requires changes in three areas: (1) people, (2) processes, and (3) technology. Installing applications focuses on only one of these areas: technology. When an organization implements a new business application it addresses issues and opportunities in all three areas.

The scope of an implementation project includes a number of business objectives:

- It ensures that the new systems are aligned with the business strategies and priorities of the organization.
- It addresses how people are going to be trained and motivated to use the technology to change the way they do their work.
- It looks at alternative ways of doing this work that leverage the capabilities of the standard software.
- It assures that the data in the system is accurate and complete.
- It does all the change management tasks that are required to reinforce the changes that are implemented.

These results will not happen in a project that just installs software. Getting significant business benefits requires a wider scope and a more structured and, unfortunately, longer implementation project. However, as will be seen in the next section, these projects do not have to take as long as they used to.

HOW FAST IS *RAPID*?

One thing that needs to be clarified is what qualifies as a rapid implementation. Is less than a year to implement an ERP package rapid? Would three months to implement a pure-play e-business application be considered slow? The only practical answer is: it depends on a lot of different things.

If you implemented all the modules of SAP in one year at a multinational organization with a great deal of reengineering, some would surely say that is rapid. Other words come to mind, like suicidal or impossible. So, that sort of project would not even qualify for consideration for a rapid approach. However, if you took six months to implement the general leader module of J. D. Edwards in a single location, that would also not meet the definition of rapid. No hard-and-fast rules on this can be given because every organization and situation is different; but some guidelines can be provided.

If an organization is using the implementation approach and tools and techniques that are described in this book, then the following projects would be considered rapid:

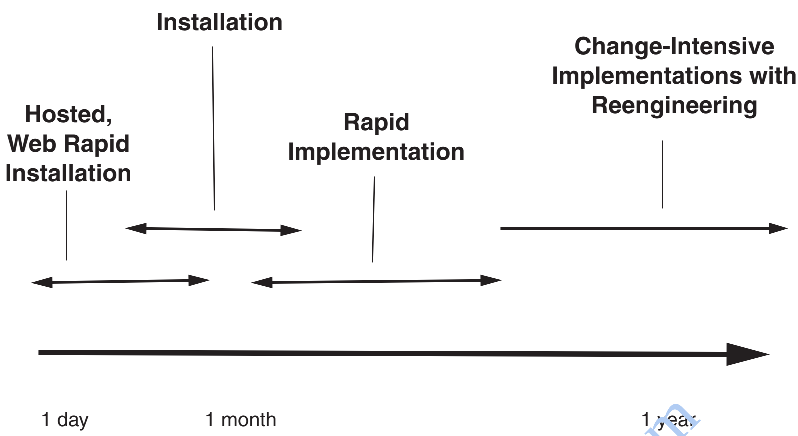
- Selection of an ERP or e-business package for the organization in one to two months
- Implementation in a midsize manufacturing company (revenues of $100 million to $1 billion) of the core modules of an ERP package (financial, sales and distribution, materials management, and manufacturing) at the corporate location and one plant, with moderate process redesign in one or two key processes, in three to six months
- Implementation in a services organization of the financial and HR/payroll modules of an ERP package with moderate process redesign in two to four months
- Implementation of a major e-business application (e.g., procurement exchange, auction site, e-retailing, B2B application) in one to three months
- Implementation of a major CRM package in an organization with a large sales force, call center, and customer service function in three to four months
- Implementation of an advanced planning and scheduling system (e.g., i2, Manugistics) in a midsize multinational distributor in two to four months

Now, even with these examples, there is room for a great deal of variation. Factors, such as the project team size, the number of end users to be trained, the quality of the data that must be converted, the speed with which decisions can be made, the politics of the organization, and the strength of the project sponsor, will all impact the time required. Instead of focusing on the actual timeframes given in these examples, think of rapid implementations as being in the two to six months' range—rather than in years. That is the type of acceleration that we are looking for in these projects.

Maybe a better way of looking at the expected speed for rapid implementations is to consider various types of implementations along a time continuum. Our expectations for projects should vary with the types of projects and the goals and objectives that have been established. This is represented in Figure 1-2.

As shown in the figure, the expected time to implement an XES application is a function of the amount of change that will occur as a result of the implementation and the technology solution used. On one end of the spectrum are projects that will result in the transformation of organizations and their processes. Most projects that do significant reengineering of business processes will still take a year or longer. It takes that long to dramatically redesign processes and do all the change management activities required to achieve the desired benefits. On the other end of the spectrum are hosted applications that are preconfigured and ready for the users to enter transactions immediately. Most applications fall between these extremes.

FIGURE 1-2 Implementation Timeframes



If you go to the web sites for the major package vendors, you will see a lot of remarkable claims. For example, at least one SAP implementation was completed in 21 days. There was also an Oracle implementation of a web-hosted financial system that was done in one day. These claims are probably accurate. However, they most likely have little bearing on how long any other implementation should take.

There are a lot of unknowns with these projects. For example, we do not know from these descriptions what was implemented, whether any historical data was converted, whether a configuration from a sister organization was cloned, or whether training materials and procedure manuals were created or used. Therefore, it is difficult to determine whether such a representation is similar to the situation your organization is facing. The best advice is to contact people who did these implementations and find out what was really done before using these statistics to set expectations for a new project.

However, to their credit, most software vendors are doing things to decrease the time it takes to implement their applications. Many are starting to use less *time to market* as a selling point. The vendors are coming out with preconfigured versions of their packages for the key industries they target. They are also working with the hardware vendors to package their software, preloaded, onto the hardware that an organization purchases to run the new system. They are developing tools and methodologies to speed up implementation tasks. In addition, the software vendors are working with their partners to make a number of application hosting options available.

The point is that there are now available tools and approaches that make rapid implementation not only possible, but proven. There are now ways to accelerate the implementation of the applications. The next section discusses the key requirements for being successful with this approach.

KEYS TO RAPID IMPLEMENTATION

Organizations have a dual challenge with rapid implementation projects: they must overcome the reasons that implementation projects failed in the past, while coming up with ways to do these projects faster. However, they are fortunate in one respect. There is general agreement on the main reasons that these projects have failed in the past. Unfortunately, the track record for most organizations is not very good for taking appropriate action to manage these known risks.

Techniques to address the normal types of project challenges are covered in Chapter 4, which focuses on how to successfully manage rapid implementations. The remainder of this chapter will spotlight challenges and keys to success that are unique, or especially important, to rapid implementations. Many of these keys to rapid implementations can be tailored for use in installation-type projects and projects that cannot be done rapidly—such as those with intensive reengineering and process redesign.

There are several keys to successful rapid implementation:

- Make quick decisions.
- Make technology infrastructure available day 1.
- Have small, cross-functional project teams.
- Do not start the clock until the team is ready.
- Use time boxing and scope management.
- Start with preconfigured versions of the software.
- Select the right package.
- Pick the right consultants.
- Take a process-driven approach.
- Conduct concurrent/parallel activities.
- Manage for speed.

Each of the keys is discussed at a high level below. Later chapters go into more detail on the specific challenges and approaches for each of these items.

Make Quick Decisions

In rapid implementations, you cannot spend the time to review, with all those affected, all the alternatives for every decision. You also cannot always work to develop a consensus for each decision with all the stakeholders before making decisions.

In XES implementations there can be literally thousands of decisions to make. SAP has over 8,000 parameter tables that must be configured for a full implementation of their product. Each of these tables has parameters that define major options for how the system will work. These tables provide the power to tailor the system to the unique needs of an organization, but they also require a lot of choices to be made. The more complex and functionally rich the standard software is, the more choices that are required. Although no organization would use all the functions in SAP, the parts that are normally used still require thousands of decisions.

In addition to configuration options, there will be hundreds of other decisions related to changes to business processes, training approaches, data conversion strategies, testing approaches and detail test plans, standard and special reports, documentation requirements, cutover schedules, and roles and responsibilities for processing with the new system. The necessity to adhere to a formal and detailed review process for a large number of these decisions will bring the project to its knees. Time is the critical resource in a rapid implementation. If the implementation team must review and justify a large number of decisions with a number of managers and end users, the timeline for the implementation will be greatly extended.

Therefore, the implementation team must be empowered to make most of the detail decisions for the implementation and use their judgment to determine when high-level decisions should be brought to the attention of the process sponsors, project sponsor, or steering committee. If the organization staffs the project team with knowledgeable, experienced, respected representatives from the functional departments, they should have the judgment to evaluate the alternatives and make good decisions for the organization. In these types of implementations, a good decision, well executed, will beat a better decision that never gets made or executed.

There will be certain decisions that require the involvement of senior management. Often these decisions center around organizational structure issues, project scope, removing project barriers, policies, capital expenditures, and decisions that will extend the length or cost of projects (other than scope). But they can also involve rollout strategies (which location gets implemented first and what is implemented there), and resolving differences of opinion among various departments that are affected by a cross-functional decision. The project sponsors and the steering committee must make all of these decisions quickly, usually within a few days.

Make Technology Infrastructure Available Day 1

The project team on a three-month implementation cannot wait six weeks to get access to the development system environment that they need to configure. They need a *sand box* environment where they can start to test different configuration options as soon as the project starts. There is a steep learning curve with most XES systems and the team must start climbing that curve immediately. Often, the roadblock is the time required by the organization to purchase and install new hardware, database management systems, and the XES software before testing by the team can begin. Or the roadblock may be that the company is still ironing out the final wording on the contract with the XES vendor, and the vendor will not send the installation CDs until all the details are resolved.

Luckily, there are ways around this problem. If the organization is using an application service provider (ASP) to do the processing for the production system, this outsourcer should be able to provide an instance of the software for use by the project team almost immediately. (As will be discussed in Chapter 3 on package selection and Chapter 7 on the IT support environment, this might be one of the key and unique requirements that will lead an organization to choose one ASP provider over another.)

Access to a development and test system may also be available from the consulting organization that is assisting in the implementation. Most of the larger consulting organizations have labs, centers of excellence, and ASP services where the XES vendor's software is already loaded and operational. One thing to check, however, is whether these organizations have the same version of the software that will be implemented by the organization—and the same operating system and database software.

Have Small, Cross-Functional Project Teams

This issue will be covered in a lot more detail in the chapter on staffing these projects (Chapter 5). However, a few key points can be made at this juncture.

First, it is important to select the right individuals to participate full-time on these implementations. The quality of the individuals assigned to the team—in terms of level, business experience, functional knowledge, talent, team attitude, ability to make decisions quickly, communication skills, and problem-solving skills—can make a significant difference in the success of the project and the ability to do these projects rapidly. These full-time team members should be the best and brightest the organization has to offer. The application cannot just be implemented by the consultants.

People from the organization should make key decisions during implementation because they will have to live with the results. Therefore, the organization's representatives on the team should take the lead in identifying issues, business alternatives, advantages and disadvantages, political factors, and recommended solutions. They provide the internal knowledge that the consultants could not be expected to have.

Second, there is a need to balance the requirement to have full-time organizational representatives from all the major functional areas addressed by the project with the desire to keep the core team small. There will be a lot of other people in the organization who will get involved and provide input during the implementation. But a rapid implementation cannot be done with a massive project team. With large teams it is difficult to have good communications among all the team members and determine and resolve integration issues rapidly. There have been ERP implementations with project teams of 400 to 500 people. However, there have been few implementations with large teams ever completed in short periods of time.

Do Not Start the Clock Until the Team Is Ready

For any implementation to be successful, it must achieve its business benefits. However, the success of a rapid implementation will also be determined, to a great extent, by how long the implementation takes. In order to give the team a fair chance at meeting the organization's expectations, it is, therefore, important for the project team to get a running start on these projects. They must be ready to make progress on project tasks as soon as the clock starts.

For this to happen, there are a number of actions that must be completed before the project officially starts. These items are prerequisites to conducting a project kickoff meeting. Activities that should occur before the kick-off—perhaps while the organization is finishing negotiations with the vendors and consultants and finalizing budget approvals—might include:

- Setting up access to a development environment
- Recruiting personnel for the team (and arranging for others to take over the team members' current jobs and responsibilities)
- Sending project team members to some initial training on the new application
- Arranging for preparation of the team work area
- Arranging for logistical support for the team
- Scheduling initial meetings with key managers and the steering committee members

If these types of tasks are not done before the project starts, the team will spin its wheels waiting for them to be completed before they can really get down to business. Unfortunately, the clock will be ticking while this occurs.

Use Time Boxing and Scope Management

Rapid implementations require *time boxing*: deciding, up front, how long the project will be allowed to take and managing the scope of the project to get it done in that timeframe. Time boxing is one of the most effective management tools to control one of the major problems in application implementation projects—scope creep. Managing scope is perhaps the key to getting the project completed on time—and getting an operational system.

A large number of ERP projects got into trouble, and were terminated, after organizations had spent large amounts of time, effort, and money. A major factor in many of these failed implementations was the inability to agree on and control the scope of the project. An indisputable fact is that an organization cannot implement all the modules of an XES package (or all the functions within complex modules) in the timeframe defined for rapid implementations. So the project's scope has to be limited in a rapid implementation.

Time boxing is also a management strategy and philosophy about application implementation. This strategy supports the position that organizations need to get benefits from their initiatives as quickly as possible—and one way to do this is to break up larger projects into a number of smaller projects. It recognizes that, for long projects, the business conditions and requirements will change before the project can be completed. It also builds on the fact that it is much easier to manage and be successful in a smaller project than in a larger one.

Time boxing is closely linked to Pareto's 80/20 rule. It says that Pareto was right; we can get 80 percent of the benefits from just 20 percent of the functions in most of the XES modules. If this is the case, why not just put in the most valuable 20 percent first? The additional functions can be added as soon as they can be justified in later projects that continuously improve business processes in response to the changing needs of the marketplace.

Start with Preconfigured Versions of the Software

A project team cannot start software configuration from scratch, with empty parameter tables, and still get implementations done in the short times allowed for rapid implementations. There is a great deal of configuration required before testing simple things, like entering an order in the test system. The team needs to start from a version of the standard system that works—and then change that version to meet the needs of the organization.

The quantity of configuration changes that will need to be made depends on how closely the initial configuration matches the needs of the organization. Some vendors only provide one default configuration: the one used for their training courses. This environment is set up to support training on all the modules in the application suite. In order to demonstrate the full range of functionality in the product, most of the training environments are for manufacturing examples. If an organization comes from the public sector, healthcare, or financial services industries, this training environment may not be a good configuration with which to begin the implementation.

Even if you are a manufacturer, there is a lot of difference between the final configurations for a discrete manufacturer (e.g., automotive) versus a company in the process manufacturing vertical (e.g., Monsanto). Therefore, manufacturers need preconfigured versions for their particular industry vertical.

Fortunately, most XES vendors tend to focus on particular industries and the unique requirements of those industries—unless they are as big as SAP and can afford to focus on all industries simultaneously (at least that seems to be the approach). The vendors market different preconfigured versions to meet the needs of specific industries. Therefore, in the selection process for the best vendor and standard package, we need to make the degree of fit with preconfigured versions of the software one of the key requirements. This leads directly to the next key to success.

Select the Right Package

There is no one application package that will meet 100 percent of the requirements of any organization, but there are several packages that will probably meet 80 to 90 percent of the requirements. Therefore, an organization should try to find the package that has the best overall fit with its unique requirements. The key and unique requirements, those things that differ from the standard processes of all organizations, should drive the selection. In general, the closer the fit, the easier it will be to implement the application using a rapid implementation approach.

There are several aspects of the package that impact the time required for implementation. When people think of package fit, they tend to focus on the functionality of the package: how well the package supports the current business processes or the *to-be* business processes.

A nonfunctional requirement might be the quality of the vendor's code. Buggier software will take longer to implement. It takes a lot of time to find and fix software bugs—even if another customer has previously identified the bug and the vendor has a software patch to fix it. Therefore, the selection process should evaluate the quality of the software and how long the newest version has been in general distribution. An organization does not want to be a test site for a new version of the software and try to do a rapid implementation at the same time.

Other things to assess during the selection process include:

- How preconfigured versions of the software fit and whether the vendor charges extra for these versions
- Whether the vendor has just-in-time, web-based training on the software (sending team members away for weeks of training is difficult during the short duration of a rapid implementation)
- Ease of changing the configuration parameters (i.e., a minimum of "load-bearing walls" that once set up, even in the preconfigured version, are difficult or impossible to change)

- Technology infrastructure required for processing and how quickly it can be acquired
- Availability of ASPs to host the processing
- Availability of accelerators (tools available to be tailored by the team so that many of the deliverables of the implementation will not have to be created from scratch)

Pick the Right Consultants

To succeed with a rapid implementation of an XES package, you need to jump-start the learning process. Without help, it takes a long time to climb the learning curve for these products and the rapid approach for implementing them. Also, the timing of these projects does not allow an implementation team to go down too many false paths or make too many major mistakes. Without experience, that is exactly what will happen in most of these implementations.

The easiest way to bypass these potential problems is to supplement the implementation team with consultants who know the package and are experienced with rapid implementation approaches and tools. Consultants should bring three skill sets to these projects:

1. Project management of rapid implementations
2. Knowledge of best practices and package-enabled process redesign approaches
3. Deep knowledge of the functionality capabilities and configuration requirements for the package that has been selected

By working side by side with the organization's team members, the consultants can keep the project moving forward in the early stages. They can transfer their knowledge to the other team members. And after the consultants leave, the organization will have knowledgeable people who can carry out future projects and enhancements of these systems.

Take a Process-Driven Approach

An XES implementation is a business project—with technical components. The technology is just a means to achieve business goals and support business initiatives. In the case of the Internet, the technology provides a way to do things that were not imaginable or feasible even five years ago. But the technology is still an enabler.

The implementation team must understand the various business processes and how they drive costs and support the strategies of the organization. This understanding is important in determining which processes will be included within the project's scope, which can be deferred to later projects, and which will be candidates for redesign.

There is not enough time in a rapid implementation to analyze in detail and redesign all the processes that are in scope. However, it usually is necessary to change some key processes in order to achieve the business benefits from the investment in the new software. So, process priorities must be established and dependencies determined to focus the work of the project team.

Process modeling will be an important activity during these projects. Process models help the team understand how business processes work and how the functions of the standard software fit into the overall process flow. The process models document the process, support its analysis, facilitate simulation of various changes, provide a means to graphically communicate the team's understanding of the process to end users and management, and serve as a basis for developing procedure and training materials.

Most of the XES vendors provide process models for their applications and tools to tailor these models to the needs of the organization. In certain cases, these process models can be used in configuring the software.

Conduct Concurrent/Parallel Activities

The traditional approach for implementing XES packages takes a serial approach to the various project tasks; some of the tasks are done in the early parts of the project and others get started halfway through or toward the end. In addition, there are major deliverables that are completed and approved, at key milestones, before the team starts any tasks in the next sections of the project.

This approach was acceptable for multiyear projects. On a two-year project you can wait until the second year to start an activity that takes only three months. However, on a three-month project, this same activity may have to be started on day one in order to complete the project on time.

Therefore, on a rapid implementation there will be more parallel and concurrent activities. A lot of activities can find themselves on the critical path. Many of the activities become more iterative in nature. This creates a need for a great deal of communication throughout the team because of the dependencies and interfaces between the activities.

Some of the activities that may go on in parallel are data conversion, training of team members and preparation of training materials for end users, test case development and testing of the configuration, procedure development, and communication and change management.

For example, ensuring the integrity of the data in the new system and getting this data converted is almost always on the critical path of a rapid implementation. Often, a great deal of work is done in data cleanup. Even though the data normally will be transferred to the new system in the month before go-live, the data cleanup must start early in the project for the organization to be prepared for the data conversion.

Manage for Speed

If the company does not have a strong, experienced project manager, the odds of having a successful rapid implementation are diminished. This project manager, usually a key manager and a representative of the user or functional areas of the organization, should have credibility with top management and the respect of the team members and key users. Ideally a skillful planner with the ability to anticipate and manage areas of risk, the project manager must be able to motivate the team members, deal effectively with personnel issues, and ensure that communication among team members and with management and end users starts early and is done often.

The project manager is responsible for managing the risks of the project. This entails anticipating all the things that can go wrong in a rapid implementation and taking action to prevent these things from occurring. Another key role of this manager is to remove barriers that prevent the completion of project activities and keep the project moving forward. A rapid implementation cannot afford to have a lot of stalls and run into a lot of dead ends. The project manager must stay on top of the status of all project activities.

The project manager must minimize major surprises on a rapid implementation because there is no time to respond to these surprises and still finish on time. For example, if one of the team members is getting behind on assigned tasks, the project manager must learn about this early so that person can be helped to get back on schedule or additional resources can be allocated to those tasks.

* * * *

By this time you should understand why it is important to develop the ability to do rapid implementations and some of the advantages of this approach. You have also seen some of the challenges to succeeding with these projects and the keys to successfully doing rapid implementations. The chapters that follow will provide a lot of information about how we actually manage and perform the activities on these projects. We start with a look at the process we use to conduct these projects: the rapid implementation roadmap.