The revolution in business and government organizations brought about by information processing technology has placed enormous burdens on — and created wide-ranging opportunities for — IS organizations. The success of government and private organizations is increasingly determined by the ability of an organization’s information system to use new technologies to meet the needs of internal and external users.

This article describes the key factors IS managers must coordinate to enable their organizations’ information systems to achieve critical goals. It identifies recent developments in organizational change and process engineering, and presents a new paradigm that IS managers can use to address the critical issues in their organizations. In doing so, it frames central questions not usually addressed by IS management paradigms. As shall be seen, the development and deployment of software in a business setting is extremely sensitive to process, culture, and organizational issues. This fact has enormous significance for IS managers who invest in and make decisions about software and related technologies.

The article makes the following assumptions:

- The need to develop or enhance an organization’s information systems is driven by the need of the business to improve services and deliver value to clients.
- IS management has the will and the determination to make the necessary changes, deploy the

**PAYOFF IDEA**
Spiraling demand for software resources means that IS managers must increasingly weigh the risks of new technology against the costs of falling behind. In this environment, successful management of the IS organization depends less on the transfer of technological implements than on adopting the characteristics of a learning organization: a mature process, a culture supportive of continuous change, and effective management.
relevant resources, and empower those who introduce and develop software products and related technologies.

Although this article focuses on applications development and maintenance, the concepts discussed relate equally to other responsibilities of the IS organization, including information technology, data center management, network management, desktop management, and IT acquisitions and maintenance.

A RAPIDLY CHANGING STRATEGIC INDUSTRY

Several large, interrelated global trends currently shape software technology and industry. One trend is the result of the numerous changes in nearly all disciplines and human endeavors that have resulted from computer-based information and control systems. These changes have matured to the point where the demand for more complex, integrated, robust, and resilient software-based systems is placing enormous demands on software development methods and technologies.

Adding to this trend are business and economic phenomena of the 1990s, including business reengineering, outsourcing, organizational change, automated flexible production, and integrated communication services, which are also increasing demand on software development capacities. The impact of these phenomena on software technology and industry has been further heightened by the maturation of other delivery technologies, including multimedia, CD-ROM, publishing on demand, interactive television, and information highways such as the Internet and the World Wide Web.

The maturation, in the last decade, of several software development technologies is another global trend. The proliferation of these newer technologies has increased decision makers’ optimism about discovering effective solutions for the software dilemma. These technologies include object-oriented methods, and several others are also becoming more effective. These will be explored in the section on software technology.

The interrelationship of these trends increases their impact on the business environment exponentially, as shown in Exhibit 1. As software development creates new opportunities for business change by providing business solutions, the resulting transformations in the business environment place larger demands on software development capacities. Changes to the existing business that results from the introduction of new software eventually reaches a level of major transformation. The transformed business also increases demand on software development capabilities because decision makers expect software development to produce new applications that support the business in its transitional state. And, once the business successfully completes the transformation, it has new demands on software development. Software development
EXHIBIT 1 — Forces Shaping the Software Development Environment

Software Development

- New Paradigm
- Change to New Paradigm Still Under Development
- Continuous Release & Development Improvements
- New Products and Technology
- New Engineering Principles

Technology Impact on Software Development

Business Impact on Software Development

Business in Steady State

- Business Solutions
- Demand for Service
- Consolidation of Process

Business in Transition

- New Applications
- Increased Demand
- Major Transformation
technologies also add to this spiraling demand by adding new products and technology that “raise the bar” on real and projected information systems performance.

To provide value to the overall enterprise, the IS organization must manage the acquisition, development, and deployment of these new technologies and software capabilities — and the changes they foster in the organization — so that it can:

• Provide state-of-the-art systems that shorten the time of business service and improve service quality, responsiveness to market, and service integration
• Manage maintenance proactively by improving it and placing it under continuous development to enable system expansion
• Reduce cost
• Provide higher quality system solutions that offer better functionality and improved reliability
• Make the information system expandable, resilient, and trustworthy for all users
• Enable creation of new business opportunities

Taken together, these economic and technological trends create demand for software resources that are orders of magnitude larger than what existed during the expansion of personal computing from 1976 to 1988. (Software resources comprise human resources, systems engineering know-how, software development tools, maintenance processes, configuration management processes, and component repositories.) As a result, IS managers face increasingly difficult decisions about balancing new software development with the business needs of the organization and its customers. They must weigh the risks of new technology against the costs of falling behind. (The software dilemma was previously mentioned.) Although much progress has been made in bringing some engineering discipline to software development and production processes, the challenge remains daunting.

THE SOFTWARE DILEMMA

Software creation and production remain largely an art. Software developer training has yet to be codified, and its certification is nonexistent. Moreover, software projects typically incur some of the worst cost and schedule overruns in the private or public sectors. For example, a milestone product like Windows 95 was released more than 18 months late and possibly more than 25% over budget. The development, publishing, and marketing budget for this product has been estimated at $1.25 billion.

Software development projects have four characteristic problems. These have existed for more than 30 years, yet still have no clear-cut solutions. The four problems are:
• Budget and schedule overruns. Software projects often exceed their schedules and budgets.
• Low quality. The quality of the software products, when released, is relatively low compared to other complex systems.
• Costly maintenance. Maintenance costs a disproportionate amount over the life span of the software. Typically, maintenance of major capital investments such as machinery and real estate is 10% of the original investment over a 10-year period. Software maintenance, on the other hand, doubles the original development costs within 3 years, making it unique among human artifacts.
• Low reusability. Typically, only a small percentage of released software products is reusable. Therefore, future developments are more likely to exceed their schedules and budgets.

History of Software Engineering

In the mid-1960s, maintenance of medium to large software systems generated approximately 75% of the total cost of the system. This phenomenon led to considerable industry research, which, in turn, led to software engineering. In simple terms, this discipline stated that software system products had to be planned, designed, constructed, and released according to engineering principles. Even then, controlling and managing schedules and costs of software projects was troublesome.

Ten years of extensive development of early structured techniques and construction tools, and several systems engineering developments, such as process and data modeling, improved the picture slightly. Some reports from the 1970s indicated that maintenance costs dropped to approximately 60% of the total life-span cost. But problems with cost and schedule control remained. The U.S. Department of Defense (DOD) reached the startling conclusion that the programming languages used to construct military software included more than several hundred dialects of more than 50 programming languages. This proliferation was the major contributor to the low reusability of systems, high maintenance costs, and low quality of system products. In the late 1970s, the DOD estimated that maintenance costs had again risen to approximately 75% of the system life span cost. It also estimated that if that pace continued, the entire DOD budget would go to maintaining software by 2010.

There were multiple software breakthroughs in the 1980s — the glorious period of the growth of small and distributed systems. This decade also witnessed development of advanced structured methods and modeling and CASE tools, and widespread deployment and use of languages such as Ada and C. This work culminated in the object-oriented era of systems engineering and construction tools at the end of the decade. The concepts of standards and process maturity also gained force during this period (see Appendix).
Effects of the Software Dilemma

The software dilemma results from the complex nature of software systems. “Software entities are more complex for their size than perhaps any other human construct because no two parts are alike.” Furthermore, the “complexity of software is an essential one, not an accidental one.”

Changing requirements, rapidly evolving technologies, changing business environments, and the demands for more integration of processes and functions are also important contributing factors.

The software dilemma has caused significant business failures. The 1990s started with numerous development challenges. Measurements of numerous actual projects revealed that:

- Medium projects, classified as 1,280 function points in size, have target completion spans of 18 months. These projects are actually finished in 32 months and have a cancellation probability of 30%.
- Large projects (i.e., 10,240 function points) usually slip from 30 months scheduled time to 60 months. The cancellation probability of large projects is approximately 50% — with a total loss of project investments.

The case-study literature of software failures is extensive, and one brief example is cited here. On January 5, 1990, AT&T’s massive long-distance network, the backbone of U.S. telecommunications, experienced a total shutdown that took more than 100 phone company personnel several hours to identify and fix. This calamity cost AT&T approximately $100 million. Estimates for what it cost the whole business community run as high as $10 billion. As the Washington Post reported, “The problem with AT&T’s software turned out to be a mistake made in just one line of a two-million-line program used to route calls. ... The flawed line, or software ‘bug,’ in the AT&T program sent the call-processing mechanism to an incorrect place in the code, where the next instruction it encountered made no sense, thus disabling the equipment.” This is not an isolated instance, and it highlights the new issues facing IS managers. As software systems become more complex, they repeat the mistakes of the major systems that preceded them. Thus, AT&T’s large system failure serves as a prototype for the present-day difficulties of IS organizations. As more organizations push to automate the production and delivery of their products and services, more IS organizations will find themselves within the realm of this type of large system failure.

To create value-added business change with software, addressing the dimension of software technology itself is important, but not sufficient. The evolution of stable, low-risk, and measurable development processes is as important as software technology itself to the production of high-quality software products and to the overall success of the organization that uses such products. In the next section, issues relating to process...
and organization in the development of a software technology capacity are reviewed.

**PROCESS, ORGANIZATION, AND CULTURE**

*Exhibit 2*, the diamond of change, is familiar to professionals who work on business reengineering. It represents the interrelated dimensions of change in an organization. Process is central to producing anything. It links organizational structure, technology, culture, and process control. As a result, a process change necessitates a change in the other dimensions, and change in the other dimensions necessitates process reengineering.

The diamond of change highlights new areas of concern for IS managers because current IS management paradigms do not address the dynamics revealed by the diamond. For example, current paradigms assume that managers have stable technology to develop applications. But, as described previously, the rate of change in software development and related technologies means that IS managers face increasingly am-

---

**EXHIBIT 2 — The Diamond of Change**

![Diagram of the Diamond of Change](image)

- **Technology**
  - Methodology (Structured Knowledge)
  - Asset Repositories

- **Process**
  - Tools and Infrastructure

- **Organization**

- **Culture**

- **Process Control**
biguous technology choices. This, in turn, means that IS managers must make increasingly difficult decisions in the related dimensions of process, organization, and culture if they are to make their IS organizations responsive to the needs of the business.

IS managers must now focus on managing the other dimensions of the organization represented by the diamond: process control, organization, and culture.

- Process control. Change itself keeps changing. It is becoming increasingly complex, as is change management. Today, business processes change with almost every project.
- Organization. Organizational structure has also entered a state of constant change. In the previous business environment, rank, position, and role were linked. Bosses typically rose through the ranks and knew the content of the work they were managing. Today’s project managers may not know the specifics of the work they manage and must draw on a different set of management skills.
- Culture. Traditional organizational theory assumes the organization has a definable culture. But because technology and related processes are in a constant state of flux, organizational culture is changing as well. The changes in technology mean that the ways people work together change continuously, often weekly.

Why should IS managers acknowledge and manage these issues? If they do not, their organizations will perish because they will not be able to respond to the needs of the business. Applications development and maintenance must deliver the goods in months, not years; and some software must be replaced every 6 months.

Even if the IS organization invests in technology or processes, the investments will not succeed unless the other dimensions are addressed as well. IS managers must manage all the dimensions of the diamond and create a dynamic balance between them. For example, investments in technology and process must be accompanied by management of the resulting changes in the culture. IS organizations that are not managed this way will not survive. They will produce unstable systems that lack adequate functionality and that will not allow the business to stay competitive. To elaborate the magnitude of these changes and to explore their impact on software technology, one must first understand the meaning of the different dimensions of the diamond of change.

The Learning Organization

Organization in the diamond of change refers to an organized structure of people and skills focused on achieving a mission or task. Organization may be defined as skilled people organized in a particular framework.
Culture is the dimension that binds people together within the organization. It includes the habits, values, constraints, inhibitions, aspirations, dreams, and motivations that drive people to do what needs or has to be done. What needs to be done is often different from what has to be done. The tension that results from this difference is an important change engine in the organization.

An organization is most dynamic when:

- Members are innovative lateral thinkers. Edward de Bono’s lateral thinking assumes that “thinking is a skill that can be improved by training, by practice, and through learning how to do better.”
- It has institutionalized knowledge acquisition and learning.
- Its structure and the culture support the difficult process of reflecting lessons learned into the organizational process, into the habits of its members, and into the thinking and expectation of its management.

In short, a learning organization continues to learn, absorbs process improvement knowledge aggressively, and constantly seeks ways to perfect its techniques and products.

Process control represents management principles, guidelines, incentives, and behavior. Management refers both to organizational management and project management. Managers must be able to handle people, resources, and time. Effective management means delivering results on time, within budget, and at the highest possible level of quality. Effective managers have more than basic skills. They feature leadership, vision, a sense of mission, the ability to manage risk, and flexibility to respond to circumstances that change constantly.

Process refers to the processes required to perform work and deliver anticipated results. A mature process is an amalgamation of many levels of processes, all of which have to be in a steady state. The way an organization achieves results and delivers value has to be documented, respected, and followed. This common process must be supported by a means of managing and incorporating change in the process itself. The guidelines for change must be derived from feedback of results, measurements of performance and delivery, and lessons learned. Achieving this state of continuous learning and process improvement is what is meant by balancing the dimensions of the diamond of change. Exhibit 3 is a schematic of mature business and improvement processes. When processes — shown as rectangles in the Exhibit — are stable, repeatable, and documented, a process is mature.

With regard to technology, it can be assumed that a learning organization, with a mature process and effective management, must possess proven technologies to deliver its results.

What do process, organization, and culture have to do with managing the IS organization? The short answer is: everything. This is the key to de-
Developing or acquiring software solutions that improve the organization’s bottom line. In today’s business environment, where the demands for software solutions are equaled only by the risks of undertaking such solutions, the management of process, organization, and culture in the IS organization is crucial to the overall success of the organization.

**Industry Guidelines.** Successfully managing the IS organization depends less on the transfer of technological implements than on the adoption of the characteristics of a learning organization; specifically, a mature process, a culture supportive of continuous change, and effective management. In other words, managing the IS organization involves more than buying packages, programming languages, or hardware. It involves implementing and following the concepts of the learning organization.

The U.S. Air Force, in collaboration with Carnegie-Mellon University of Pittsburgh, developed a model that can help organizations measure maturity in their software development processes, project management, and
organization capability. The model, known as the Software Engineering Institute's Capability Maturity Model (SEI-CMM), was unveiled in 1991. Although the SEI-CMM does not explicitly promote the concepts of the learning organization, it overlaps considerably with these concepts. The model provides five maturity levels and several key practice areas within each maturity level. These practice areas can be assessed within an organization or a project. The underlying thesis of the model is that the higher the maturity level of the software process, the lower the risk of failure, which is defined as failure of the system product to meet costs, meet schedules, and produce high-quality system products. All organizations are assumed to be compliant with level 1. An organization is mature from the SEI perspective when it is able to document process knowledge and use that knowledge consistently, which is also a key characteristic of the learning organization.

A similar effort in the international and commercial arena was launched by the International Organization for Standardization when it extended the ISO 9000 standards to cover software production. Quality standards and process standards were proposed, again to help lower the risks of failure and improve product quality.

Today, there is a strong move in the United States to make software builders and contractors compliant with SEI-CMM and ISO 9000. The U.S. Air Force has mandated that all its software developers reach level 3 of the SEI-CMM by 1998. NASA is considering a similar policy. As of September 1994, 261 organizations had been assessed; 75% of these organizations were stuck in level 1, while 24% were at levels 2 or 3. Only two projects — not organizations — earned the highest rating of level 5.

In summary, compliance with the SEI model and ISO makes it much more likely that the organization is a learning organization and promotes higher levels of systems engineering practice maturity.

RECOMMENDED COURSE OF ACTION

This article uses the applications development environment within the IS organization to illustrate general changes currently facing IS managers. The main thesis is that these changes demand a radical shift in the IS organization's management paradigm. This shift entails moving from a paradigm focused on management of resources, skills, and technical implements in a hierarchical structure to a paradigm that promotes a flat organization of knowledge workers who continuously learn and adapt new methods, technologies, and disciplines. This new paradigm not only fosters a state of perpetual learning in the organization, but also strong management ownership of and involvement in the operation and management processes. This paradigm shift must be accompanied by use of process measures to enhance systems engineering practices and enable IS managers to master increasingly complex systems and services.
IS managers should no longer consider technological solutions in isolation; the realities of the technology and the marketplace demand that such solutions be implemented as part of a larger process of rethinking and improving process, organization, and culture.
Appendix

EVOLUTION OF SYSTEMS ENGINEERING

Systems engineering is made up of disciplined methods and techniques that pertain to the structure of system components, the functionality of each component, and the interaction between components. Systems engineering usually includes requirements engineering, also known as analysis; system architecture with layering and services; distribution schema; software design; workflow design; and the logical design of databases. Software engineering includes software metrics, modeling, methods, and techniques associated with the conception and detailing of the system before it is constructed. Prototyping, though it is dependent on tools, is included in the software engineering domain as well.

A methodology is a structured, integrated approach to the engineering of software and of the business. The methodology usually provides a coherent definition of components, activities, tasks, steps, work products, tips, techniques, and guidelines for achieving the development of software. The kernel of a methodology is a thought discipline referred to as systems engineering. Classifications of systems engineering vary in the literature. A common approach is to use milestone technologies as the departure point for the evolution of the technology. A different classification scheme is used here to present the evolution of systems technology. This classification scheme focuses more on the set of ideas, practices, and techniques that prevailed in each school of thought. The classification is depicted in Exhibit 4 and a brief description follows.

Artisan
The artisan approach prevailed in software development from the beginning of software engineering until the late 1960s. This school used limited modeling tools. Text was the only means of describing requirements or design aspects. Often, the process for software development processes was composed of a limited planning stage and then a long construction or coding stage. The construction stage incorporated the creation of the program codes as well as the creation of the architecture, design, and testing activities. Systems were created with limited or no documentation. Metrics to measure performance or create estimates were rarely used.
Early Structured
With the early structured school, software systems practice took a giant step toward creating the foundations of an engineering discipline. Semantically richer models were introduced. This stage started in the mid-1960s and lasted until the end of the 1970s. In this stage, text remained the strongest descriptive tool. However, practitioners recognized that system documentation had to be created before the coding of the system. During this stage, analysis was also recognized as a phase within the development life cycle. Analysis techniques, however, were limited. This stage also distinguished between the design of the system and the design of the code. Systems architecture was not recognized as a concept, except toward the end of this stage, when real-time system practice began to use the concept of systems architecture, which was borrowed from hardware practices. Metrics were formal but limited. They were mostly a function of lines of code.

Advanced Structured
Numerous systems concepts were tested and developed from 1975 through the 1980s. The experience with early structured methods yielded a wealth of knowledge about systems engineering. The portfolio of sys-
tems modeling techniques was greatly expanded. The set of modeling techniques now included event modeling, information modeling, data modeling, and process. Architecture was recognized as an important stage in the development of software systems. Analysis was enhanced with formal integrated essential models that provided views of the systems from several perspectives. Design was formally delineated from construction. The construction concept was expanded beyond coding, to coding and assembly of components. Documentation became a design vehicle and documented the system. Metrics were formally recognized and some techniques were associated with metrics, such as the function point technique.

Object Oriented

Object-oriented concepts have existed in experimental programming languages since the early 1970s. These concepts matured into formal programming languages in the early 1980s. Some of the concepts were developed in engineering and special application systems. Interest grew by the mid-1980s with the expansion of graphical user interfaces and other developments in programming environments. By the late 1980s, a systems engineering approach to object orientation began to take shape.

Object-oriented methods involved the expansion of the modeling tool sets. Scenario building and verification promoted further development. Object-oriented concepts such as inheritance, assembly, and polymorphism encouraged the development of reusable component libraries. Object libraries, object containers of other objects, and object databases now hold out the promise that some effective approach for reusable components might finally be at hand.

It is difficult to acquire systems engineering thinking and practice. They require continuous training and continuous learning from the lessons of both failed and successful projects. The essence of this process is a learning organization. Practitioners cannot acquire systems engineering practice by completing training and doing a few projects. There must be a systematic learning process that changes systems engineering thinking and improves the process.

Notes
3. Ibid.