Communications of the Association for Information Systems

Volume 7 Article 17

10-25-2001

Which Life Cycle--Work System, Information System, or Software?

Steven Alter
University of San Francisco, alter@usfca.edu

Recommended Citation
Alter, Steven (2001) "Which Life Cycle--Work System, Information System, or Software?," *Communications of the Association for Information Systems*: Vol. 7 , Article 17.
DOI: 10.17705/1CAIS.00717
Available at: https://aisel.aisnet.org/cais/vol7/iss1/17

Communications of the Association for Information Systems

Volume 7, Article 17
October 2001

WHICH LIFE CYCLE - - - WORK SYSTEM, INFORMATION SYSTEM, OR SOFTWARE?

Steven Alter
School of Business and Management
University of San Francisco
alter@usfca.edu

RESEARCH

WHICH LIFE CYCLE - - - WORK SYSTEM, INFORMATION SYSTEM, OR SOFTWARE?

Steven Alter
School of Business and Management
University of San Francisco
alter@usfca.edu

ABSTRACT

This article presents the work system life cycle (WSLC) model, according to which a work system, an information system, or a software product passes through one or more iterations of four phases: initiation, development, implementation, and operation and maintenance. Although this descriptive model is both clear enough to understand readily and specific enough to apply easily, it encompasses a variety of other models commonly used to describe information system life cycles, organizational change processes, projects, and the life cycles of software products. The explicit inclusion of both an operation and maintenance phase and iterations allows it to cover both continuous and discontinuous change. This article explains the need for this type of model and shows how it spans more than a dozen descriptive and normative models that appear in the IS literature.

The WSLC model could help bridge the communication gap between business and IT professionals, could help both do their own system-related work, and could help students grasp the broad alternatives for building and modifying systems. Comparison of the WSLC phases with the phases of other models shows that many of those models might be misleading to people who are not primed to understand why their goals and assumptions are quite different. Both

software-focused and organization-focused models in the IS literature tend to emphasize development activities and de-emphasize implementation in the organization and ongoing operation and maintenance. Instead of providing clarity, use of some of these models in practice and teaching might reinforce common misunderstandings and naive expectations that stereotype information system projects as either IT projects performed by technologists or as organizational change projects in which technology plays only a secondary role. The WSLC model encourages a balanced view that includes both the organizational and technological viewpoints without minimizing either.

Keywords: work system, information system, system life cycle, system development life cycle, project management, project life cycle, information system education

I. INTRODUCTION

This article defines a particular life cycle model for work systems and argues that it encompasses most existing life cycle models for systems, processes, and projects. Here are some of the situations in which this model might be valuable:

- Programming manager M has had difficulty communicating with business colleagues. Part of the problem is the lack of a shared framework for talking about the past, present, and future of specific systems. A mutually understandable life cycle model might help.

- Banks N and P merged six years ago and decided to use P's ATM network. A major overhaul occurred three years ago. Now the merged bank wants to extend the ATM network to support additional financial products from two partner firms. A broadly applicable life cycle model might help in understanding major problems that occurred in the past and in charting a course that might be more successful in the future.

- Instructor R read Markus's admonition that "in-house IS specialists in IT-using companies are no longer building and maintaining internal systems the way we describe and prescribe in our textbooks" because custom-developed software is increasingly being replaced by ERP software purchased from vendors and configured by conducting a gap analysis that starts with the capabilities of the software. [Alter et al, 2001, p. 26] Instructor R wants a good way to compare alternative life cycle models to avoid misleading students and to illuminate difficult issues and tradeoffs in current practice.

- Researcher S wants to follow Orlikowski and Barley's [2001, p. 158] call for "more interplay between the fields of organization studies and information technology," and for research 'that embraces the importance of simultaneously understanding the role of human agency as embedded in institutional contexts as well as the constraints and affordances of technologies as material systems." A broadly applicable life cycle model might help in comparing case studies focusing on topics such as path dependence, diffusion of IT innovation, and management of IT-enabled change.

Each of these situations involves a need to understand several *different* views or versions of system life cycles. In the first case, the different views involve IT professionals and business professionals. In the second, the issue is how to start making sense of a complex situation involving a particular system's past, present, and future. The third calls for an easily understandable way to organize and explain topics such as performing projects in general, building and maintaining custom-programmed software, implementing commercial ERP packages, reengineering business processes, and building Web sites and other specialized types of information systems. The fourth calls for a way to

conceptualize a system's past in terms of more than just organizational theory or just IT development.

This article defines a particular life cycle model for work systems and argues that it encompasses most existing life cycle models for systems, processes, and projects. This model is both clear enough to understand readily and specific enough to apply easily. If it truly satisfies these criteria it could help in communication between business and IT professionals, in understanding the history and future of specific systems, in providing instruction to students, and in promoting further cross-fertilization between IS research and organizational studies.

This article is part of ongoing research attempting to produce a reasonably brief, cogent set of ideas that are genuinely applicable to most information systems and that can be used effectively by typical business professionals (and IT professionals) as they participate in developing, using, and improving information systems. Two previous *CAIS* articles [Alter, 1999 and 2001] argued that the fundamental concepts of information systems are mostly about work systems. Those ideas have been applied as the basis of a systems analysis method designed to help business professionals use business terms to understand and evaluate systems and proposed system improvements [Alter, 2002, Chapter 2]. One direction for extending these ideas involves better ways to describe, characterize, and evaluate both existing systems and various types of overlap and other relationships between systems. The current article takes a different tack by focusing on system life cycles.

The current article extends and clarifies a general life cycle model that was introduced in the previous articles. After defining a work system and summarizing previous discussions of why information systems and projects are special cases of work systems, this article introduces the work system life cycle (WSLC) model, an iterative four-phase model that applies to any work system, and hence to any information system or project. These phases are:

- initiation,
- development,

- implementation, and
- operation and maintenance.

To express the fact that many systems go through major revisions, the model's operation and maintenance phase includes a recurring decision about whether to continue maintenance and incremental improvements (continuous change), whether to undertake a major revision by starting a new iteration of the four phases (discontinuous change), or whether to terminate the system.

The broad applicability of the WSLC model is demonstrated by showing how it can be used to outline the life cycle of information systems built through a variety of methods including custom programming, software packages, and iterative development and implementation (Section III). Next is an exploration of how the WSLC helps in visualizing the emphasis and limitations of a number of descriptive and normative life cycle models that have appeared in the IS literature. Comparison with the WSLC model shows that most of these life cycle models focus on aspects of the development phase and de-emphasize or ignore implementation in the organization and ongoing operation and maintenance of the resulting system (Section IV). Different life cycle models reflect views of what is important or interesting in the development phase. Some are clearly about projects in which software, documentation, training materials, and other artifacts are acquired or created, modified or configured, and tested (Section V). Others emphasize process redesign and almost seem to take software development for granted (Section VI). The concluding section (Section VII) argues that the comparison of the various life cycle models demonstrates the applicability and usefulness of the WSLC model. It also discusses implications for curriculum and for communication between business and IT professionals.

II. DISTINCTION BETWEEN WORK SYSTEM, INFORMATION SYSTEM, AND SOFTWARE

Previous *CAIS* articles explained why the concept of "work system" is a useful basis for understanding and analyzing information systems and other systems in organizations. [Alter, 1999 and 2001]. This concept of work system is

more general than information system, but less general than the broader concept of system. The concept "work system" reduces the generality of "system" by providing more focus, thereby creating a genuinely useful way of looking at most systems in organizations. (Readers familiar with the treatment of these concepts in previous *CAIS* articles should skip to Section III.)

The term "work system" was used in some of the socio-technical systems literature of several decades ago but may not have been defined clearly enough or used rigorously enough to be popularized as a central concept for understanding information systems.[1] As defined in the previous articles, a work system is a system in which human participants and/or machines perform a business process using information, technology, and other resources to produce products and/or services for internal or external customers. A work system operates within a surrounding context and typically relies on infrastructure that may not be owned or controlled by work system participants or their managers. [Alter, 1999] Typical business organizations contain work systems that procure materials from suppliers, produce products, deliver products to customers, find customers, create financial reports, hire employees, coordinate work across departments, and perform many other functions.

Based on this definition, a work system can be summarized in terms of eight elements that should be included in even a superficial understanding of a work system (which might be an information system or a project in an organization). The

- *business process*,

[1] For example, Mumford and Weir [1979, p. 3] speak of "the design and implementation of a new work system." Similarly, Davis and Taylor [1979, p. xv] mention "attempts at comprehensive work systems design, including the social systems within which the work systems are embedded." More recently, Land [2000] says "socio-technical methods focus on design of work systems to improve the welfare of employees. The prime aim of redesigning work systems is the improvement of the quality of working life." I was not aware of this previous use of the term "work system" by socio-technical researchers when writing my first *CAIS* article about work systems, and even mentioned a series of Internet searches that found a variety of uses of this term, such as "back-to-work systems", but few uses in the context of understanding or designing systems in organizations. [Alter, 1999, p. 13], The subsequent inclusion of PDF files in the universe covered by the Google search engine made it much easier to find examples of this usage through subsequent searches in July 2001.

- *participants*,
- *information*, and
- *technology*

constitute the system performing the work. The work system's outputs are the

- *products and services*

received and used by its

- *customers*.

Including the products and services and the customers among the elements even though they are not part of the system reflects the notion that a work system exists to produce products and services for internal or external customers. Including the

- *context* and
- *infrastructure*

reflects the impact of external factors on system operation and success.

Information systems are work systems in their own right since they consist of human participants and/or machines performing a business process using information, technology, and other resources to produce products and/or services for internal or external customers. Information systems are a special case of work system in which the business process is devoted to only six types of activities: capturing, transmitting, storing, retrieving, manipulating, and displaying data. For example, information systems cannot deliver packages, manufacture refrigerators, or perform surgery even though they may be important parts of work systems that do these things. The purpose of most information systems is to support one or more work systems. Although information systems and the work systems they support were often quite separable decades ago when most business computing was still card-based and batch-oriented, today many

important information systems overlap significantly with the work systems they serve. For example, the information system could be a small part of a much larger work system; it could encompass most of a work system devoted to processing information; it could be shared in various ways among multiple work systems that may or may not be directly related.

Projects are a special case of work system because they also they consist of human participants and/or machines performing a business process using information, technology, and other resources to produce products and/or services for internal or external customers. The unique characteristic of projects is that a project's business process is designed to end with a planned termination step after which project resources are released and the project ceases to exist.

Software is neither an information system nor a work system. It is a part of the technology used in a computerized information system. Of the many types of software, the only type of software discussed in this article is application software whose features and capabilities are meant to be directly noticeable to information system participants (who are software users). This software ranges from general-purpose office software, such as Microsoft Word and Excel, through situation-specific software, such as an SAP or Oracle ERP module designed for a particular business function in a particular industry. We will ignore middleware, operating systems, and other software that should be transparent to information system participants and that might be changed without their noticing.

The foregoing distinctions among work systems, information systems, projects, and software are a necessary prelude to the following discussion of life cycle models. Although the IS literature contains a number of life cycle models related to work systems, information systems, projects, and software, it is sometimes unclear about which models pertain to which topic.

III. THE WORK SYSTEM LIFE CYCLE MODEL

The *CAIS* articles that explained the concept of work system also discussed a general, four-phase life cycle that applies to work systems. That

model is explained in detail in Alter [2002]. The updated version in this article adds explicit recognition of the iterative nature of many system life cycles, thereby incorporating the common distinction between continuous and discontinuous change. Since the term "work system life cycle" will be used frequently and distinguished from other life cycle models, the new model is abbreviated WSLC in this article.

The WSLC model assumes that the life cycle of a system in an organization consists of one or more iterations of four phases that apply whenever a project creates or significantly changes a work system (which may be an information system). The four phases in a given iteration include initiation, development, implementation, and operation and maintenance. Different types of situations, such as projects employing different methods for building or implementing systems, might call for different steps within each phase. Regardless of whether the topic is an information system or some other type of work system, the operational system that results from the four phases in an iteration might be called the nth iteration of the system or the nth generation or the nth major revision. For example, the third generation of a firm's sales system might be the result of a project that includes eliminating several sales offices, installing and configuring version 7.5b of a vendor's software, changing sales procedures, and providing extensive sales training.

The fact that a system life cycle might involve multiple iterations is a key difference between a system life cycle and a biological life cycle posed in terms of predictable periods of birth, development, maturity, decay, and death. Assuming that a firm does not die within the time interval of interest, its basic systems such as systems for hiring people, manufacturing products, and servicing customers also do not die, but rather, go through a series of iterations that are usually not predictable in advance. Each iteration typically ends when a period of continuous change (maintenance and incremental improvements) gives way to a discontinuous change in which major parts of the system are re-thought, modified, and possibly merged with other systems.

This distinction between a biological life cycle that includes predictable decay and termination versus a system life cycle that could extend through many iterations raises questions about the meaning of the term "life cycle." For example, consider some of the different types of entities whose life cycles have been discussed in the IS and management literature:

- projects, which are designed to produce something and terminate
- software products, which might be designed to survive through many major revisions
- information systems, which might exist indefinitely even though technologies and procedures might change (e.g., the Vatican Library or the U.S. census system
- organizations, (e.g., IBM's sales organization) some of which are temporary but many of which are not designed to go out of existence even if they do change shape occasionally.

This article assumes that the term life cycle applies to any of these situations and refers to a broad outline of the typical or desired developmental path of whatever type of entity is being considered. The phases in the WSLC therefore encompass many different types of life cycles that involve a range of diverse business processes.

Figure 1 illustrates the WSLC model and shows that the output of each phase within an iteration is an input to the next phase. It also uses italics to show some of the reasons for returning to a previous phase to correct errors and omissions and to exploit opportunities that become apparent later in the project.

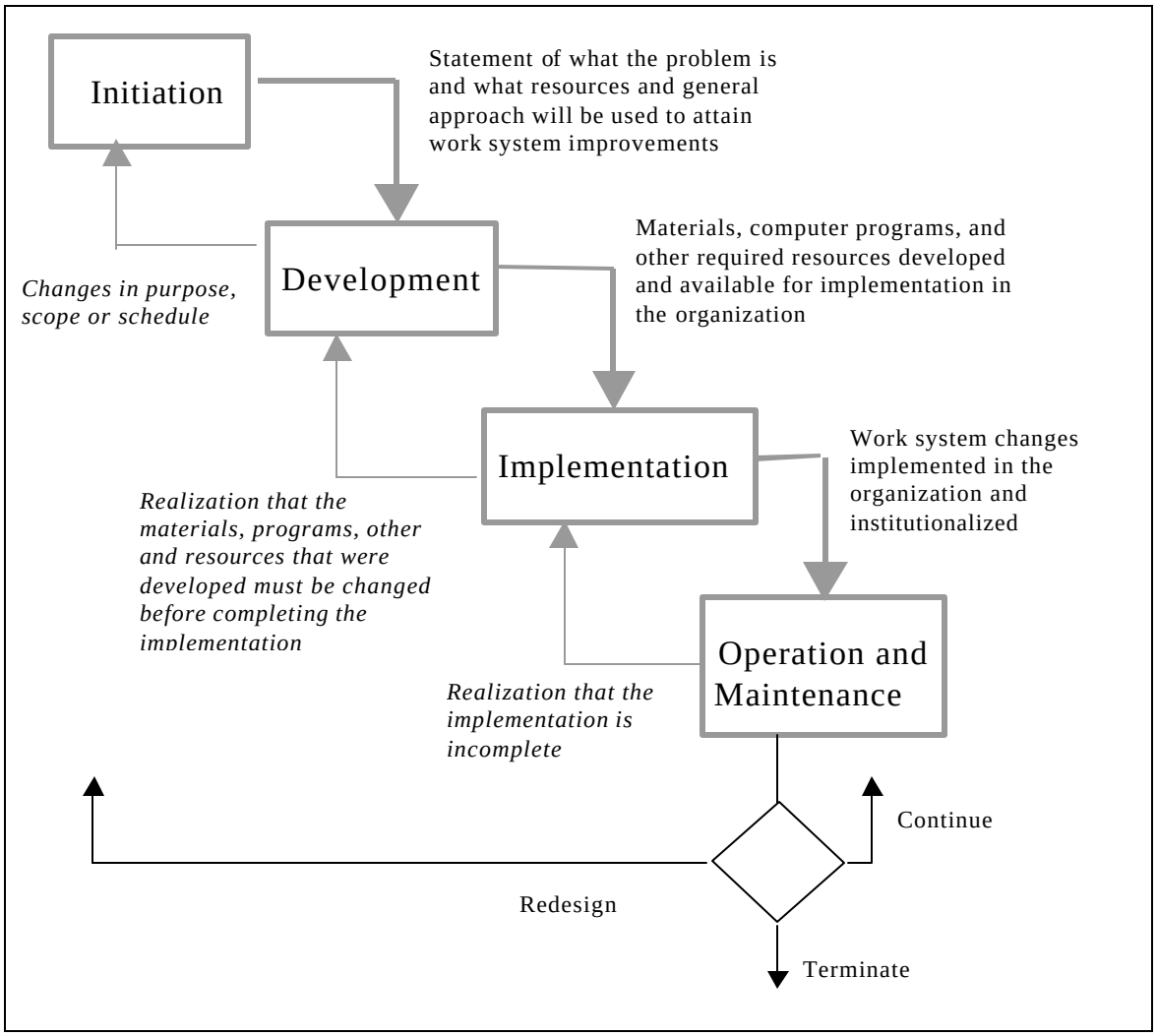

Figure 1. The Work System Life Cycle (WSLC) Model

At a superficial level the four phases in the WSLC model may seem obvious because effective ongoing operation of a system in an organization implies that:

- someone must have wanted the system in first place (initiation)
- people had to design and create the system (development)
- people had to make it operational (implementation)
- people had to keep it alive (operation and maintenance)

Whether or not the WSLC model is obvious in retrospect, it was not obvious to me around 1991, when I was writing the first edition of Alter [2002] and trying to define a "common denominator" that would provide insight about different ways to build information systems. To help reduce confusion often generated by techno-centric views of projects and systems I tried to express the phases without using IT terminology. After all, the purpose of the effort is to change a work system, not just to produce computer programs that operate correctly. In addition, focusing on a work system life cycle rather than an IS life cycle might support more effective communication between business professionals and IT professionals, many of whom have been trained to focus on a software life cycle, and to see projects in IT-centric terms. For example, instead of viewing *implementation* as the process of making a work system change operational in a organization, many programmers view it as the process of designing and programming software and are ready to declare victory when the software operates on a computer consistent with its requirements. (This programming-oriented rather than business-oriented view of implementation is implied by the widely publicized *IS '97 Model Curriculum Guidelines for Undergraduate Degree Programs in Information Systems*. [Davis et al, 1997] Of its ten courses, IS '97.8 is "Physical Design and Implementation with DBMS" and IS '97.9 is "Physical Design and Implementation with a Programming Environment.')

Whether or not to include organizational implementation and operation and maintenance in project or life cycle models is a surprisingly common issue in the IS field. For example, Strassmann closes a chapter about IT investment politics by advising a CIO to "commit only to schedule, systems capabilities and information services budgets. Everything else is beyond a CIO's capability to explain or prove." [Strassmann, 1997, p. 247] This may be valid advice for some CIOs, but if the IS field focused on no more than schedules, capabilities, and budgets it would miss a lot of insight about common problems and perverse results. For example, consider some of the problems that are recognized after "successful" completion of software development projects:

1. All planned functionality was developed within budget and schedule but the information system is considered a disappointment or failure because implementation of the revised work system is ineffective.

2. Despite attempting to attain "technology-enabled change," no significant change occurs and work continues to be done the same way with the new technology. An example is an ongoing ERP implementation in which the software is being configured in a way that minimizes business process changes, leaving some managers wondering why they went through all the effort to create data integration when the organization apparently intends to continue operating the same business processes within the same functional silos.

3. In some situations, technology adoption results in less effective work systems. For example, even in ERP projects that generate significant corporate benefits, local work systems may become less effective because the ERP software cannot support unique local processes or because the ERP software was not configured properly.

My guess is that problems such as these probably account for a nontrivial part of the productivity paradox.

MORE ABOUT THE PHASES OF THE WSLC MODEL

Since the WSLC will be used as the basis of comparing other life cycle models, it is useful to say a bit more about each of the phases.

Initiation. The initiation phase is the process of clarifying the reasons for changing the work system, identifying the people and processes that will be affected, describing in general terms what the changes will entail, and allocating the time and other resources necessary to accomplish the change. This phase

may occur in response to obvious problems, such as unavailable or incorrect data. It may be part of a planning process searching for innovations even if current systems pose no overt problems. When the work system uses software, errors and omissions in this phase may result in software that seems to work on the computer but needs expensive retrofitting after initial attempts at implementation in the organization. Unless the initial investigation shows the project should be dropped, this phase concludes with a verbal or written agreement about the proposed system's general function and scope, plus a shared understanding that it is economically justified and technically and organizationally feasible. Depending on the situation, this agreement might be general and informal, or might be quite specific in identifying budgets, timelines, and measurable objectives. Key issues in this phase include attaining agreement on the purpose and goals of the proposed change and making sure that the likely benefits far exceed the likely costs in terms of time and resources. The larger the project the more desirable it is to document specific expectations along with a plan for accomplishing genuine results (as opposed to just performing specific activities at specific times). Regardless of how formal the agreement is, the details of the desired changes will be worked out the development phase.

Development. The development phase is the process of defining, creating, or obtaining the tools, documentation, procedures, facilities, and any other physical and informational resources needed before the change can be implemented successfully in the organization. This phase includes deciding how the work system will operate and specifying which parts of the work will be computerized and which parts will be manual. The term development is used in most projects that create software, but it is also sometimes used in ERP installations. For example, one book on SAP projects says that the "development stage" of those projects includes refining the vision, configuring the prototype, and technical development work such as interfacing, data conversion, and customization [Doane, 1998, pp. 70-78].

Completion of development does not mean "the system works." Rather, it only means that the tools, documentation, and procedures have been produced

and that computerized parts of the work system operate correctly on computers. Whether or not the computerized parts of the work system actually work adequately will be determined later by how the entire work system operates in the organization. Key issues in this phase revolve around creating or obtaining all required resources in a cost-effective manner and, if necessary, demonstrating that tools and procedures actually meet the requirements. Completion of this phase means that the tools seem to function properly. Whether the work system will absorb or reject the desired changes is determined by the next phase.

Implementation. The implementation phase is the process of making the desired changes operational in the organization, which in the case of e-business might be a virtual organization involving a number of different companies. Implementation activities include planning, training of work system participants, conversion to the new work methods, and follow-up to ensure the entire work system operates as it should. Ideally, the bulk of the work in this phase should occur after development is complete, meaning that all tools and procedures are ready and that all software has been tested and operates correctly on the computer. This phase ends when the updated work system operates effectively in the organization.

An initial step in this phase is detailed planning for the conversion from the old way of doing things to the new. After work system participants are trained, the actual conversion to the new work system occurs. This step usually raises issues about how to convert to a new process with minimum pain and how to deal with political questions and changes in power relationships. In all of this, success of the computerized parts of the work system is determined partially by features and partially by the development and implementation process itself. The likelihood of success drops if this process cannot overcome the inertia of current business processes or if the implementation itself causes resistance.

If a work system's development phase created or modified an information system, some parts of the conversion involve the changeover to the new or modified information system and other parts of the conversion may be changes in practices that are unrelated to the information system. When the conversion

affects data and methods used for transaction processing, it is often necessary to perform each transaction twice during the conversion, once using the old work system and once using the new work system in order to minimize the risk that the new work system will have unforeseen problems that jeopardize or prevent its successful operation.

Operation and maintenance. This final phase involves keeping the work system operating effectively by monitoring its performance and making minor changes that do not require a major project. When an information system plays a major role in a work system, someone must make sure that it continues to operate, that it provides benefits, and that desired changes are at least considered. This phase continues until the system is terminated or until major changes are required. At that time a new iteration of the four phases starts; management allocates resources to initiate a project; the new initiation phase ends with specific ideas about what should change; the new development phase begins, and so on. Operation and maintenance may not seem as intellectually intriguing as development, but by typical estimates it absorbs the majority of a firm's information system expenses.

ITERATIONS OF THE FOUR PHASES

The earlier version of the WSLC model included only the four phases. The updated version adds a loop saying that a system's life cycle might simply traverse the four phases once or might go through a series of iterations of those phases. The diagram of the WSLC model in Figure 1 represents the trigger for the iterations as a recurring decision that surfaces occasionally during a system's operation and maintenance phase. That decision is about the future of the system and has three general options: continue operation and maintenance (plus incremental change), redesign the system in a significant way (going back to initiation), or terminate the system. In terms of the common distinction between continuous and discontinuous change, the "continue" branch in Figure 1 represents the incremental fixes and small improvements that constitute continuous change, whereas "redesign" and "terminate" represent the discontinuities.

The desirability of adding the iteration loop became apparent to me after I read a paper [Cox et al, 2001] submitted by a team of students attending the University of San Francisco's Professional MBA program. Based on current plans plus recollections and company memos going back to the early 1990s, this paper looked at the history of a "data distribution system" at an engineering firm with five geographically dispersed offices. The students were familiar with the four phases, but their way of describing the system's history applied the phases in four successive iterations, which they called sneakernet and modems, local area networks, private wide area network, and virtual private network. Their paper summarized what happened in each of the phases in each of the iterations, how the data distribution system supported specific work systems, and why each successive iteration was undertaken. Their portrayal of the history as a sequence of iterations not only made sense, but also led me to think about other situations such as ERP implementations that significantly modify previous versions of work systems in areas such as marketing, finance, production, purchasing, and human resources. It is certainly important to look at ERP implementations as projects, but a longer-term view looking at the situation as yet another iteration of one or more work systems might provide additional insights.

IV. HOW THE WSLC PHASES ENCOMPASS ALTERNATIVE INFORMATION SYSTEM LIFE CYCLES

The previous section defined the WSLC model and explained its basic terminology, which includes:

- Life cycle: Broad outline of a typical or desired developmental path of a type of entity such as a work system, information system, project, or software product.
- Iterations: A system life cycle consists of one or more iterations of four phases.
- Four phases: initiation, development, implementation, operation and maintenance

- Steps within each phase: Typical or desired activities within a particular phase. The steps in a phase such as development differ depending on the type situation.

This section will show that the four phases of the WSLC model (without including iterations for major revisions) encompass five idealized life cycle models for information systems:

- custom programming from detailed specifications
- configuring packaged software
- iterative prototyping
- phased implementation
- evolutionary development.

Using the WSLC model to recognize and compare the alternatives starts to address the issue of providing students with a realistic (and organized) view of the range of methods for building information systems. The subsequent sections will extend this idea by showing how the WSLC model can be used to compare a variety of life cycle models related to projects, software development (rather than information systems), business process design, and organizational change. In each instance the discussion is brief and is mostly concerned with summarizing the phases and limitations of the various models. In general, this article is much less concerned with model details, such as whether programming in one model is subdivided into program design, coding, and testing in another. It also treats the various idealized models as separate even though any real world situation would call for defining a life cycle model that combined the most pertinent features of the various idealized models.

The broad applicability of the WSLC model in summarizing so many different life cycle models stems from the fact that every one of these situations can be viewed as a work system in its own right. In other words, since the WSLC model applies to any work system, it also applies to projects in general, software development, ERP implementations, process redesign, and organizational change, all of which are work systems.

In this section, the table or figure associated with each of five idealized life cycle models emphasizes differences between the current model and the models mentioned previously and does not repeat activities and characteristics that most have in common, such as the fact that it is sometimes necessary to rework previous phases. Also, note that each idealized model represents only a single iteration rather than the multiple iterations included in the WSLC model.

CUSTOM PROGRAMMING FROM DETAILED SPECIFICATIONS

The "custom programming from specification" life cycle emphasizes carefully controlled custom programming from a detailed specification. This life cycle is sometimes called the system development life cycle (SDLC) or the structured life cycle, but this article will not use those terms because it will look at a number of different life cycles that are at least somewhat structured and that involve system development. The "waterfall" version of this life cycle (discussed later) is more or less a benchmark whose characteristics and shortcomings motivate the other approaches.

Table 1 uses the four phases of the WSLC model to present one version of the major steps of a life cycle characterized by custom programming from detailed specifications. Different versions of this life cycle name these steps differently and combine or divide them in various ways, but the basic idea is still the same, namely, define exactly what is required from the programmers and maintain tight control over the project by dividing it into a series of steps that create specific deliverables. Ideally, requiring that each step produce the appropriate outputs on time and within budget should assure success. In reality, this life cycle model involves so many steps, deliverables, and sign-offs within each phase that it guarantees a lengthy project even without unplanned complications and delays. In some cases, the real requirements change before the new work system is implemented.

Table 1. Major Steps in an IS Life Cycle Involving Custom Programming from a Detailed Specification

Phase in the WSLC Model	*Major Steps During This Phase*
Initiation	· Feasibility study · Project planning
Development	· Detailed requirements analysis (external design) · Internal design · Hardware acquisition and installation · Programming and unit testing · Documentation · Integration testing
Implementation	· Implementation planning · Training · Conversion · Acceptance testing · Post-implementation audit
Operation and Maintenance	· Ongoing operation and support · Maintenance · Recurring decision: continue, redesign, or terminate

The steps in Table 1 provide a useful starting point for thinking about almost any life cycle model. As will be seen throughout this article, many life cycle models either downplay or ignore the steps during the implementation and operation and maintenance phases that must occur before an information system can have a significant impact. The inclusion or omission of particular steps during the development phase is also noteworthy. For example, assume that no formal internal design step is performed or that no documentation step is performed, as might happen with a very small information system developed by an end user. The resulting information system might be quite useful and might succeed in its own terms, but the lack of documentation and careful internal design would probably affect subsequent efforts to extend its scope. Similarly, an IS life cycle model that does not include programming is certainly possible when vendor software is used, but the mere fact that the software was programmed elsewhere does not eliminate either the possibility that further programming is needed or the importance of the programming and testing techniques that the vendor used.

CONFIGURING PACKAGED SOFTWARE

The "configuring packaged software" life cycle involves purchasing or renting software instead of building it, but still requires a high level of effort across the entire life cycle. Unique aspects of this life cycle mentioned in Table 2 show why purchasing or renting the software eliminates most custom programming but still leaves many time- and resource-consuming steps during the development phase. Some group still needs to select the package, configure it, and possibly customize it. Some group still needs to produce documentation and training material that fit the current situation rather than the situation in which the vendor might have produced the documentation. Table 2 also shows that reliance on a software package provided by an external vendor has ramifications during the implementation and operation and maintenance phases.

Table 2. Unique Aspects of an IS Life Cycle that Relies on Packaged Software

Phase in the WSLC model	*Unique aspects related to using packaged software*
Initiation	· Decide to purchase software instead of producing it.
Development	· Select the vendor and software package · Configure the software package · Where necessary, customize the programs · Modify the vendor's documentation and training material to fit the situation
Implementation	· Train users on new concepts and terminology required by the vendor's software package · Explain and obtain buy-in whenever specific work systems operate less effectively with the new software (in order to provide benefits elsewhere)
Operation and Maintenance	· Try to influence the vendor to produce new releases whose features are beneficial in this situation · If vendor software was customized or if add-ons were programmed for this site, maintain the customizations and add-ons across new vendor releases.

ITERATIVE PROTOTYPING

The "iterative prototyping" life cycle uses multiple iterations of a prototype during development because it is difficult or impractical to define the requirements in enough detail. Using a prototype helps in visualizing the difference between the way work is done currently and the way it will be done when the new information system is in place. Although a prototype may serve as a sanity check in other life cycle approaches, multiple iterations of prototypes is often associated with less structured situations, such as building decision support

systems and reporting systems. Although sometimes effective as a way to understand the requirements, iterating through a series of prototypes may place unusual strains on both the programmers and the user representatives. Table 3 identifies some of the unique aspects of using iterative prototyping.

Table 3. Unique Aspects of an IS Life Cycle that Relies on Iterative Prototyping

Phase in the WSLC Model	*Unique Aspects Related to Using Prototypes*
Initiation	· Decide that it will be necessary to use iterative prototyping.
Development	· Determine requirements by using multiple iterations of a prototype. · Decide whether to implement the prototype in the organization or to redesign the programs to create a more robust computer system before starting implementation
Implementation	· Implementation may be easier because the prototype proved the concept and demonstrated that key assumptions were correct.
Operation and Maintenance	· If the prototype was not re-programmed, operation and maintenance may be more difficult because the computer system is not as well designed and documented.

PHASED IMPLEMENTATION

Regardless of whether the development phase produces software or acquires and configures it, a "phased implementation" life cycle divides the implementation into a series of smaller implementation projects in order to reduce risks and to identify and use lessons from the first part of the implementation. This type of phased implementation is practical only where it is possible to convert to a new work system one part at a time or where it is possible for part of an organization to convert to a new or revised work system before other parts of the organization move to the same system. Dividing the implementation into a series of smaller implementation projects delays the benefits from having the entire new system in operation but reduces the likelihood of a major implementation fiasco. A variation on this approach is what Fichman and Moses [1999] call "results-driven incrementalism." With this approach, organizational implementation of software functionality is divided into a sequence of "business releases" each of which introduces a subset of the software that is capable of supporting a particular, trackable organizational

change. Figure 2 represents phased implementation as a series of separable implementation projects that occur after the development phase.

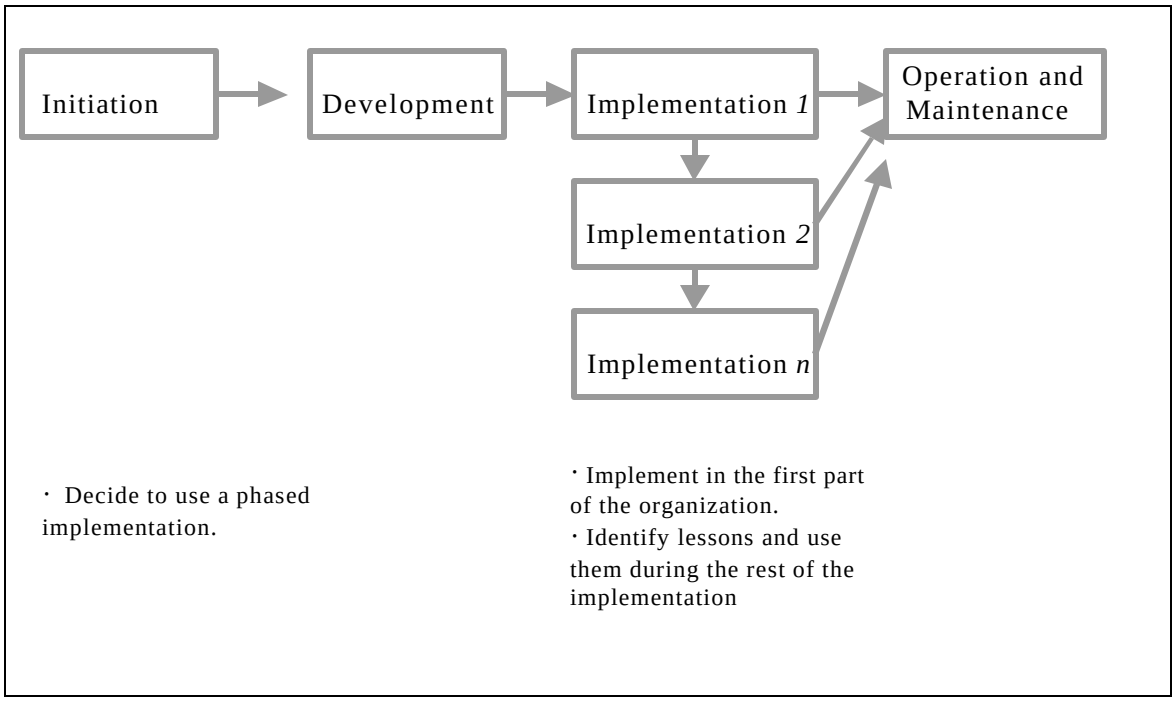

Figure 2. Unique Aspects of Using a Phased Implementation

EVOLUTIONARY DEVELOPMENT

The "evolutionary development" life cycle breaks a larger project into a series of smaller projects each of which goes through development and implementation phases. Figure 3 represents this life cycle as a series of small, separable development and implementation projects. Ideally, the use of a succession of smaller projects should reduce the risk of completing development without receiving implementation-related feedback. If the initial implementation reveals unanticipated problems, the use of an evolutionary approach minimizes the disruption and makes it easier to go back to the previous way of doing the work. An evolutionary approach also reduces risk by basing each successive layer of development and implementation on the success of the previous layer. In addition, at least some of the benefits should be attained more quickly.

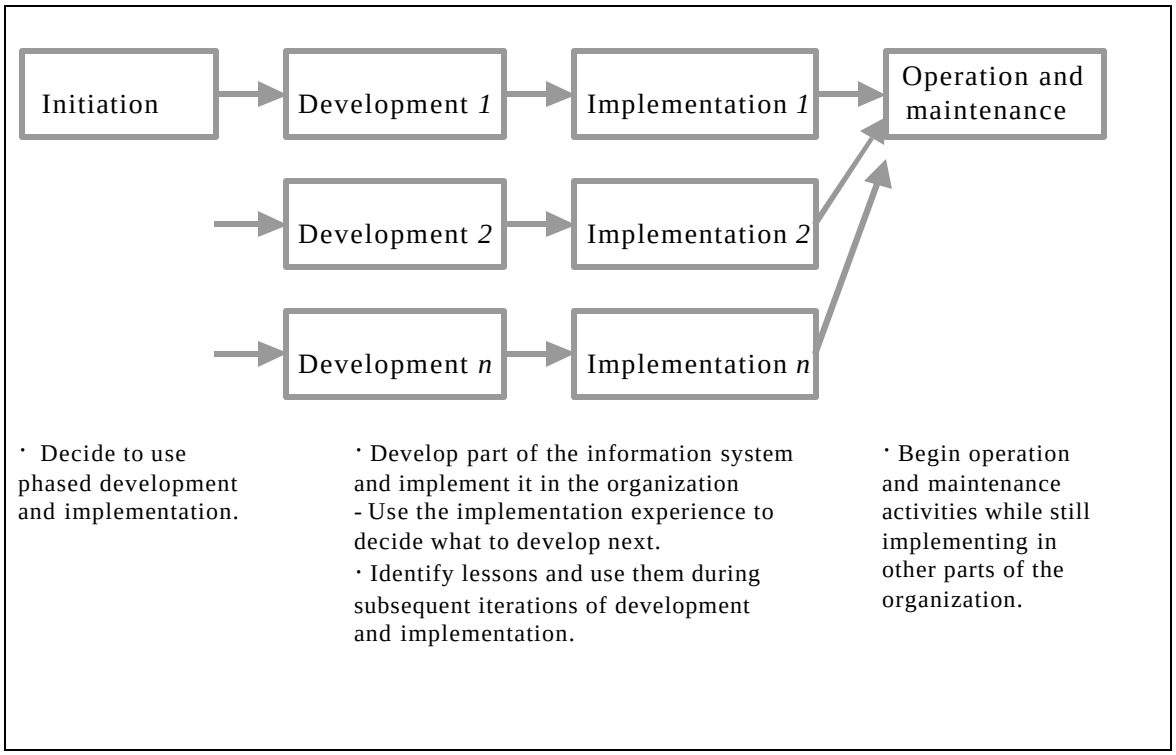

Figure 3. Unique Aspects of Using Evolutionary Development

A variation described as "an improvisational model for change management" assumes that technology-based innovations might not always be planned in advance, such as when an organization's applications of Lotus Notes and other groupware tools evolve over time. According to an improvisational model, tools such as these might be exploited through multiple cycles of

- anticipated change,
- emergent change, and
- opportunity-based change.

The anticipated changes are those that are planned in advance. The emergent changes arise spontaneously from local innovations that are not anticipated or intended. The opportunity-based changes are not anticipated in advance but "are introduced purposefully and intentionally during the change process in response to an unexpected opportunity, event, or breakdown." [Orlikowski and Hofman, 1997, p. 13] In a broader context, Truex, Baskerville,

and Klein [1999] note that the stability assumed by traditional IS development goals does not exist in emergent organizations in which "culture, meaning, social relationships, decision processes and so on are continually emergent, following no predefined pattern." In such organizations they call for revoking traditional IS development goals and moving toward goals based on the assumption that systems should be under constant development, can never be fully specified, and are subject to constant adjustment and adaptation."

V. LIFE CYCLE MODELS FOCUSING ON PROJECTS THAT BUILD OR INSTALL TECHNICAL TOOLS

The previous section introduced the WSLC model and showed that its four phases spanned five idealized life cycle models for information systems. This section and the subsequent section look at a number of other life cycle models that are not specifically about information systems but are viewed as part of the IS literature. Some of these models are mostly about projects that build or install software; others are about organizational change or process redesign but don't pay much attention to how any required software will be created, modified, or acquired. Table 4 lists the models that will be covered and shows that the project-related models appear in the current section and the other models appear in the subsequent section. These models were chosen as examples illustrating the range of topics and terms included in life cycle models in the IS literature. Other models might have been selected, but an equally broad range of models probably would have covered most of the same topics.

Table 4. Life Cycle Models Covered in the Remainder of this Article

Life Cycle Models for Projects that Build or Install Software	*Life Cycle Models that Emphasize Organizational Change or Process Redesign*
· a life cycle model for a typical project · a waterfall model for software development · Microsoft's synchronize and stabilize model · a corporate web site development methodology · project phases for a major SAP R/3 implementation	· the Lewin-Schein change model · Mumford's ETHICS model for socio-technical systems · Harrington's business process improvement model · Davenport's process innovation model · Kettinger et al's stage-activity framework summarizing business process reengineering models used by different consulting companies

The coverage of each of these models includes a table organizing its phases and steps using the four WSLC phases. Brief comments about each model characterize its coverage, limitations, or other features. This sequence of summaries demonstrates that each model emphasizes some aspects of projects, software development, business process reengineering, or organizational change, but downplays or ignores other aspects that most IT and business professionals should be aware of. For our purposes, the issue is not that two models have slightly different names or combinations for related steps, but rather that most of the models devote scant attention to at least some issues that are essential for attaining benefits.

LIFE CYCLE MODEL FOR A TYPICAL PROJECT

The life cycle model shown in Table 5 was presented in a *CAIS* tutorial on a manager's view of software project management [Jurison, 1999, pp. 8-10] and represents a typical project that might confront a software development manager. The cells of Table 5 next to the WSLC phases of implementation and operation and maintenance are empty because this model ends before implementation in the organization begins. Noting that this end point is not universal, Jurison says "for IS projects, the execution phase frequently extends beyond delivery of the end product and includes system implementation, the process of putting the system into operation in the client's organization. It is not uncommon to have system implementation handled by a separate project team because the implementation team often must function as a change agent rather than as a developer. System implementation introduces a new set of project management challenges that are beyond the scope of this tutorial." [Jurison, 1999, p.9]

Implementation and operation and maintenance are legitimately beyond the scope of Jurison's tutorial, but they clearly are within the scope of an information system life cycle if the information system is to have any impact whatsoever. Thus, a project manager's view of a life cycle model for a software development project might be quite different from that same manager's view of the entire life cycle for an information system.

Table 5. Life Cycle for a Software Development Project

Phase in the WSLC Model	*Generic Phases of a Project [Jurison, 1999]*	*Steps Related to Each Phase [Jurison, 1999]*
Initiation	Conceptual	· Identify needs · Establish goals · Determine feasibility · Prepare proposal · Estimate time and resources (rough) · Identify key people · Get approval
	Planning	· Prepare plans · Develop budget · Develop schedule · Assemble project team · Build and test prototypes · Get approval for next phase
Development	Execution	· Perform work · Procure material · Build and test · Verify performance · Modify as required
	Termination	· Transfer responsibility · Release resources · Transfer team members · Reward people · Conduct review
Implementation	x x x x x	x x x x x
Operation and Maintenance	x x x x x	x x x x x

WATERFALL LIFE CYCLE FOR SOFTWARE DEVELOPMENT

The waterfall life cycle [Boehm, 1981] is a classical software development model that is cited frequently. In effect, it applies a general project management life cycle to organize software development tasks. According to the waterfall metaphor, each phase should be completed and its deliverables approved before the next phase commences. Table 6 uses the WSLC model to compare two summary versions of the waterfall model [Jurison, 1999; Cusomano and Selby, 1997]. The sequence of phases in the two versions is consistent but some of the specific terms differ. As with the general project life cycle model in Table 5, the waterfall models in Table 6 emphasize the development phase of the WSLC and

seem to assume that implementation and operation and maintenance in a specific organization are mostly beyond the boundaries of the problem, or at least are less interesting.

Table 6. Two Recent Versions of the Waterfall Life Cycle Model

Phase in the WSLC Model	*Phases in Waterfall Life Cycle [Jurison, 1999]*	*Phases in a Waterfall Life Cycle [Cusomano and Selby, 1997]*
Initiation	· Feasibility study	· System requirements
Development	· Requirements analysis	· Software requirements
	· Design	· Preliminary program design · Analysis · Program design
	· Programming	· Coding
	· System testing	· Testing
Implementation	· Implementation	x x x x x
Operation and Maintenance	x x x x x	· Operations

Many citations to the waterfall model use it as a basis of comparison for explaining alternative approaches that recognize the frequent necessity to revise previous assumptions and understandings as project participants dig into the details and as requirements change due to external events. For example, the "spiral model" [Boehm, 1988] is a widely cited modification of the waterfall life cycle that calls for rapid iterations of development and implementation, thereby converting it to an evolutionary development approach (see Figure 3).

MICROSOFT'S "SYNCH-AND-STABILIZE" MODEL

Table 7 summarizes a description of the software development approach Microsoft uses (or at least used in the mid-1990s) to produce successive releases of software products. Cusomano and Selby [1997] use a comparison with the waterfall model (Table 6) to explain the unique aspects and advantages of Microsoft's model. They label it as a "synch-and-stabilize approach whose logic is to synchronize what people are doing as individuals and as members of parallel teams and periodically stabilize the product in increments as a project proceeds, rather than once at the end of a project." They say that Microsoft employees refer to their techniques as the "milestone," "daily build," "nightly

build," or "zero-defect" process. [Cusomano and Selby, 1997, p.54]. Other software vendors use a variety of life cycle models for producing software releases. One such variation is the "evolutionary-delivery" model [McCormack, 2001] whose additional wrinkle involves dividing the upcoming release into "microprojects," each of which receives user feedback after a cycle of feature design, coding, and integration testing.

Table 7. Microsoft's Synchronize and Stabilize Method

Phase in the WSLC Model	*Phases in Microsoft's Method [Cusomano and Selby, 1997]*	*Steps in Each Phase in Microsoft's Method [Cusomano and Selby, 1997]*
Initiation	Planning	・Vision statement ・Specification document ・Schedule and team formation
Development	Development	(Each team develops features in 3 or 4 sequential subprojects) ・Managers coordinate evolution of the specification ・Developers design, code and debug. ・Testers pair with developers for continuous testing
	Stabilization	・Internal testing of the entire release ・External testing through beta sites ・Release preparation
Implementation	x x x x x	x x x x x
Operation and Maintenance	x x x x x	x x x x x

Although different software vendors use different methods for producing software releases, an important commonality among them is that their life cycle methods reflect goals that differ from those of internal IS groups: "External software product developers try to provide generic products that users can easily adapt to their unique requirements, whereas the traditional in-house approach is to tightly tailor the software to the users' unique requirements and to provide them with little flexibility in how to use the software." [comments by Markus in Alter et al, 2001, p. 29]. Table 7 also says that a release is completed when it ships, i.e., when development is finished. Microsoft tracks bugs, trouble reports, and other aspects of the their products in use, but as with the general project model (Table 5) and the waterfall model (Table 6), implementation in the

customer organization is the customer's problem. Obviously Microsoft needs to provide tools and techniques that make it reasonably easy to install and use their software, but most of the typical issues in the last two phases of the WSLC model are beyond the scope of the synch-and-stabilize model.

CORPORATE WEB SITE DEVELOPMENT METHODOLOGY

The IS literature includes a number of specialized life cycle models related to specific types of information systems. A recent example is the corporate Web site development methodology proposed by Sherrell and Chen [2001]. They say that "most organizations do not have a formal process for Web site development" and that a corporate Web site methodology should be "applicable to any type of corporate Web site (intranet, Web-presence, transactional, or extranet)." They define "the W software life cycle model" together with an associated methodology for corporate Web site development (CWSD). The W model is so named because its graphical representation is in the form of a W, with planning, requirements analysis, and system design along the left diagonal, incremental implementation at the middle vertex, and maintenance, system testing, acceptance testing, and maintenance along the right diagonal. The steps in CWSD correspond to the phases of the W software life cycle model. The purpose of the model is "to furnish Web developers with an overall framework that describes the required phases in constructing and maintaining corporate Web sites. The aim of the methodology is to reduce the pitfalls of corporate Web site construction and to increase the completeness, compatibility, and quality of resulting sites." [Sherrell and Chen, 2001, p. 4]

The summary of the CWSD model in Table 8 is an example of how the term *implementation* has different meanings in different life cycle models. The steps associated with the "incremental implementation" phase in the CWSD model involve programming and testing the Web site before it goes into use. The minimal attention to implementation in the WSLC sense raises questions about whether achieving benefits is or should be included in the model. For an intranet or an extranet, as for any other information system, the benefits occur only after an implementation effort (in the WSLC sense) changes the way people do some

type of work within a company (for an intranet) or within a virtual organization that may include many companies (for an extranet). Without this type of implementation, the intranet or extranet probably won't have much impact. A similar comment applies for transactional Web sites, which also encounter the types of system development and debugging issues that any transaction processing system encounters. Thus, the WSLC model helps in seeing that the CWSD methodology is mostly about creating a Web site, and de-emphasizes organizational implementation and operation and maintenance issues.

Table 8. A Corporate Web Site Development Methodology

Phase in the WSLC Model	*Phases of the W Software Life Cycle Model [Sherrell and Chen, 2001]*	*Numbered Steps in the Corporate Web Site Development (CWSD) methodology [Sherrell and Chen, 2001]*
Initiation	Planning	1. Identify Web site project
		2. Conduct feasibility study or initial interview
		3. Form project team
Development	Requirements analysis	4. Outline overall structure of the Web site
		5. Filter information and refine requirements.
		6. Compare with Web site standards
	System design	7. Hold pre-design meetings
		8. Develop prototype
	Incremental Implementation	9. Construct site using builds - digitize materials and design art work - code programs - unit test and integrate - validate with customer
	System testing	10. Install and test
Implementation	Acceptance testing	11. Deliver web site
Operation and Maintenance	Maintenance	12. Maintain web site

PROJECT PHASES FOR A PARTICULAR SAP R/3 IMPLEMENTATION

Table 9 presents a slightly simplified version of Table 3 in a case study of the implementation of SAP R/3 at NIBCO, a $460 million manufacturer of valves and pipefittings. [Brown and Vessey, 2001]. The vendor, SAP, developed the

software through its own development process, but the case focused on how this software would be implemented at NIBCO. Even though the software was developed elsewhere, the initiation, development, and implementation phases of the WSLC model absorbed a great deal of time and effort at NIBCO. The case focuses on the 15 months devoted to these phases, and ends on Dec. 31, 1997, just as the new system is going live (i.e., being implemented) in the organization.

Table 9. Phases and Major Activities in the SAP R/3 Implementation at NIBCO

Phase in the WSLC Model	*Phase of the NIBCO SAP Project [Brown and Vessey, 2001]*	*Major Activities in the NIBCO SAP Project [slightly revised from Table 3 in Brown and Vessey, 2001, p. 26]*
Initiation	(before "preparation" phase)	· Long range planning study · Selection of SAP · Selection of consultants · Decision to attempt a "big-bang" implementation
Development	Preparation	· Final project plan – scope and cost. · "As-is" business analysis. · Technical infrastructure specifications. · Project management and tracking tools developed.
	Analysis	· Document "as-is" processes as "to-be" processes. · Analyze gap between "to-be" processes and R/3 processes. · Identify process improvements and changes to fit R/3. · Document inputs, outputs, triggers, business activities, roles, change categories, training requirements.
	Design	· Configure R/3. · Develop training materials. · Develop specifications for master data, external interfaces, and reports · Develop and review prototypes for modules and processes
Implementation	Implementation	(Some overlap with design phase) · Data cleanup. · Determine customization needed across plants. · Address outstanding hardware issues · Plan transition to new system. · Develop post-live support processes. · Determine customization needed across plants.
Operation and Maintenance	x x x x x	x x x x x

The activities covered by the case involve a great deal of preparation for ongoing operation and maintenance, even though the actual operation and maintenance phase is not covered because this phase has not yet occurred as of the end of the case.

Of greatest interest for our purposes are the preparation, analysis, and design phases that correspond to the development phase in the WSLC model. Notice how the activities in these phases devote a great deal of detailed attention to analyzing requirements and deciding how the software can be configured to best suit the requirements. At NIBCO these activities cost millions of dollars and absorbed the attention of a large number of managers. These initial investments are certainly a far cry from a vendor's view of a software life cycle that is mostly about iterations of software development.

VI. LIFE CYCLE MODELS THAT FOCUS ON PROCESS AND ORGANIZATIONAL CHANGE

A blossoming of literature from the management and organizational change community in recent decades has paralleled the blossoming of software development and maintenance literature from the IT community during the same period. For example in 1961, Benne, Bennis, and Chin co-authored and edited *The Planning of Change*, a book that Bennis looks back upon as "an attempt to encompass in one volume the most seminal and original essays in the yet unborn field of organizational change." [Bennis, 2000, p. 235] Since 1961, many authors offered guidelines for how leaders should act and think and also about the factors and methods that are associated with successful organizational change. We will look at several models that emphasize life cycle phases or activities that occur during process redesign or organizational change.

LEWIN-SCHEIN CHANGE MODEL

The Lewin-Schein change model is used frequently in the IS implementation literature. As summarized in Table 10, this model says that any significant organizational change goes through three stages, unfreezing, moving,

and refreezing.[2] [Lewin, 1951] Unfreezing occurs prior to and during the initiation phase in the WSLC model. Moving encompasses the things that happen during development and implementation. Refreezing should happen during the operation and maintenance phase as the new ways of doing the work become ingrained. Table 10 shows that the middle stage, moving, combines the development and implementation phases of the WSLC model.

Table 10. Lewin-Schein Change Model

Phase in the WSLC Model	*Lewin-Schein Stage*	*Activities During this Stage*
Initiation	Unfreezing	Creating an awareness of the need for change and a climate of receptivity to change
Development	Moving	Changing the forces and behaviors that define the initial situation, developing new methods, and learning new attitudes and behaviors
Implementation		
Operation and Maintenance	Refreezing	Reinforcing the changes that have occurred, thereby maintaining the new work practices

Characterizing an organizational change process in just three words might seem simplistic, but the three words make an important point that is not apparent in the software-oriented life cycle models. Consider an ERP project that did not go well. Millions of dollars were spent, the software was configured and installed on the computer, some parts of the organization are using some of the installed features but others are still using previous manual or computerized systems for record keeping. The Lewin-Schein model helps in seeing what might have gone wrong. Perhaps the unfreezing stage never happened for many potential users. Perhaps they were never convinced of the need to change from their local status quo. Yes, the ERP effort might help someone elsewhere in the organization, but that might not make it personally worthwhile to change existing processes that operated effectively until changes elsewhere interfered. For these potential users, the moving stage and the desired changes in work practices never really occurred even though the technical staff installed and configured the software on

[2] In subsequent discussions of process consultation, Schein re-cast the second stage as

the computer and someone provided training. Wherever the change did not occur refreezing could not occur because the new work practices were never in place.

The Lewin-Schein change model is certainly not the only general organizational change model in the literature, and many of the other models seem to be embellishments of its basic points. For example, Nadler [1998, p. 75] uses a five phase model for leading discontinuous change:

- recognizing the change imperative,
- developing a shared direction,
- implementing the change,
- consolidating the change, and
- sustaining the change.

The first step corresponds to unfreezing, the next two involve moving, and the last two involve refreezing. For our purposes the main point is that organizational change clearly involves much more than designing a process, even though design seems to be the main concern of the models that will be discussed next.

THE *ETHICS* METHOD FROM SOCIO-TECHNICAL WORK DESIGN

Table 11 summarizes the ETHICS (Effective Technical and Human Implementation of Computer Systems) method developed by Mumford and others associated with the socio-technical approach [Mumford and Weir, 1979]. When the paper was published, the ETHICS method was still being developed but was "sufficiently far advanced to be of practical assistance to organizations introducing new systems of work or improving old ones, and who wish to try to use this opportunity to improve the job satisfaction of their employees. … Briefly, the ETHICS method consists of a set of steps which must be taken in the design and implementation of a new work system. At each stage, technical and human needs are taken into account, so that the system is designed specifically to meet both technical and human objectives at one and the same time." Table 11

"changing through cognitive restructuring." [Schein, 1987, p. 93]

combines information from two sources. [Mumford and Weir, 1979 and Hirschheim and Klein, 1994, pp. 106-107, pp. 26-43] to summarize this method in a manner similar to this article's other summary tables

Table 11. The ETHICS Method

Phase in the WSLC model	*Stage in ETHICS [Hirschheim and Klein, 1994, pp. 106-107; Mumford and Weir, pp. 26-43]*	*Activities Related to these Steps [Hirschheim and Klein, 19 94, pp. 106-107; Mumford and Weir, pp. 26-43]*
Initiation	Diagnosis	· Define the problem and its boundaries · Analyze the current system · Identify key objectives and tasks · Identify information needs for the design. · Identify and rank efficiency needs and job satisfaction needs
Development	Socio-technical design	· Identify social objectives, resources, and constraints · Identify technical objectives, resources, and constraints · Match social and technical alternatives · Select best socio-technical solution
	Set out alternative solutions	· Specify social and technical alternatives · Identify which social and technical alternatives are compatible
	Select best socio-technical solution	· Rank compatible alternatives in terms of costs resources and constraints · Select the best alternative
Implementation	x x x x x	x x x x x
Operation and maintenance	System monitoring	· After the change, monitor to make sure objectives continue to be valid
	Post change evaluation	· Verify that "post-change fit" is better than "pre-change fit". · Take remedial action if necessary.

Although the acronym ETHICS stands for Effective Technical and Human Implementation of Computer Systems, the summarized version of the method in Table 11 focuses on designing processes that balance social and technical objectives, resources, and constraints. In contrast to the software development models discussed earlier, the summary of the ETHICS method (which is admittedly an interpretation based on two particular sources) does not say anything about software per se. It is also interesting that the ETHICS method (again, as interpreted here) addresses the steps in the WSLC implementation

phase only indirectly by making sure that process of redesigning the system (within the WSLC development phase) considers both social and technical factors and by including system monitoring and post change evaluation.

The ETHICS method makes important points about involvement and job satisfaction, but it and other socio-technical methods are used much less than advocates had hoped. Although the socio-technical approach "has been applied in many industrial environments, frequently as a way to enrich the jobs of assembly-line workers or to introduce technology effectively into unionized work environments, … this approach has stalled over the last decade, because the assumption that more satisfied workers would become more productive workers has not always been realized." [Davenport, 1993, p. 317]. At an IFIP 8.2 conference in June 2000, Mumford started her paper, "Socio-technical design is an enigma. It has offered so much and produced so little … When it was first developed after the Second World War and was seen by its developers as a means for optimizing the intelligence and skills of human beings and associating these with new technologies which would revolutionize the way we live and work. This did happen for a while in the 1970's when many industries tried to implement socio-technical methods of working. But initiatives gradually faded away so that today, …we still have many people working on jobs which are routine, tightly controlled and provide few opportunities for personal development." [Mumford, 2000].

THREE MODELS RELATED TO PROCESS REDESIGN AND REENGINEERING

Table 12 uses the WSLC model to compare the major phases in three process-oriented models developed in the 1990s. The authors of these models describe them as models about business process improvement, process innovation, and business process reengineering, respectively. In contrast with the software and project models, these models emphasize the analysis and redesign of the process that will be changed and say much less about software development. Each model starts with something like the WSLC initiation phase,

says a lot of about the design aspects of development, and then covers implementation and operation and maintenance to a limited extent.

Table 12. Comparison of Three Process-Oriented Life Cycle Models

Phase in the WSLC Model	*Phases of Business Process Improvement [Harrington, 1991]*	*Major Step in Process Innovation [Davenport, 1993, p. 25]*	*Stage in BPR Framework [Kettinger et al, 1997]*
Initiation	.Organize for improvement	· Identify processes for innovation · Identify change levers	· Envision · Initiate
Development	.Understand the process Streamline the process	· Develop process visions · Understand existing processes · Design and prototype the new process (includes migration and implementation activities)	· Diagnose · Redesign
Implementation	x x x x x	(included in phase above)	· Reconstruct
Operation and maintenance	.Measurements and controls .Continuous improvement	x x x x x	· Evaluate

Tables 13, 14, 15 show the activities within the phases of each of these models. The reason for presenting all three, instead of just one, is to show that process and organizational change models do address more of the work system life cycle than software-oriented models, but that they tend to say less about software development. This finding is not surprising, but it illustrates the range of current alternatives for models that might be used (effectively or ineffectively) for activities including software development, process and organizational change, communication between business and IT professionals, and teaching business and computer science students.

Table 13. Business Process Improvement [Harrington, 1991]

Phase in the WSLC Model	*Phases of Business Process Improvement [Harrington, 1991]*	*Summary of Many Activities Related to Each Step in Process Improvement [Harrington, 1991, pp. 21-22]*
Initiation	Organize for improvement	· Appoint a champion for process improvement and train executives · Develop an improvement model · Review business strategy and customer requirements · Select critical processes and appoint process owners
Development	Understand the process	· Produce a process overview and flow diagram · Define process scope, mission, boundaries, and expectations · Collect cost, time, and value data · Perform process walkthroughs · Update process documentation
	Streamline the process	· Identify improvement opportunities · Define changes that eliminate bureaucracy and activities that don't add value, simplify the process, reduce cycle time, eliminate errors, standardize, and automate parts of the process. · Document the new process
Implementation		Train employees
Operation and maintenance	Measurements and controls	· Develop process measures, targets, and a feedback system · Audit and assess the costs of quality problems
	Continuous improvement	· Qualify the process and perform qualification reviews · Define and eliminate process problems Benchmark the process

Table 14. Process Innovation [Davenport, 1993]

Phase in the WSLC Model	*Major Steps in Process Innovation [Davenport, 1993, p. 25]*	*Activities Related to this Step in Process Innovation [Davenport, 1993, pp. 27, 48, 51, 120, 139, 154]*
Initiation	Identify processes for innovation	· Enumerate major processes · Determine process boundaries · Assess strategic relevance · Judge process health · Qualify process culture and politics
	Identify change levers	· Identify technical and human opportunities for change · Identify potential human and technical constraints and determine which will be accepted. · Identify human and technical enablers.
Development	Develop process visions	· Assess business strategy · Consult process customers for objectives · Benchmark for performance targets · Formulate process objectives · Define desired process attributes
	Understand existing processes	· Describe the current process flow · Measure and assess the process in terms of new objectives · Identify problems and short term improvements · Assess current IT and organization
	Design and prototype the new process	· Brainstorm design alternatives · Select preferred alternative based on feasibility, risk, and benefits
Implementation		· Prototype the new process · Develop a migration strategy · Implement new organizational structures and systems
Operation and maintenance	x x x x x	x x x x x

Table 15. Stage-Activity Framework for Business Process Reengineering [Kettinger et al, 1997, p. 61]

Phase in the WSLC Model	*Stages in BPR Framework [Kettinger et al, 1997]*	*Related Activities in the BPR Framework [Kettinger et al, 1997]*
Initiation	Envision	· Establish management commitment and vision · Discover reengineering opportunities · Identify IT levers · Select process
	Initiate	· Inform stakeholders · Organize reengineering teams · Conduct project planning · Determine external process customer requirements · Set performance goals
Development	Diagnose	· Document existing process · Analyze existing process
	Redesign	· Define and analyze new process concepts · Prototype and detailed design of a new process · Design human resource structure · Analyze and design information system
Implementation	Reconstruct	· Reorganize · Implement information system · Train users · Process cut-over
Operation and maintenance	Evaluate	· Evaluate process performance · Link to continuous improvement programs

The model in Table 13 appears in *Business Process Improvement* [Harrington, 1991], which presents a detailed method for improving business processes. Compared to the models described earlier, this model is more concerned about assuring the desired change in the organization will actually happen. For example, the "organize for improvement" phase includes appointing a process improvement champion and appointing process owners for critical processes. Subsequent steps call for process-related training at various levels. On the other hand, like the ETHICS model, Harrington's model says little about any software that might have to be developed and about the effort of converting from the previous process to the new process.

In *Process Innovation, Reengineering Work through Information Technology*, Davenport [1993] adds IT to the discussion and says explicitly that the process innovations he is looking at involve IT. Table 14 shows that the major step "understand existing processes" includes the activity "assess current IT and

organization." Similarly, the major step "design and prototype the new process" includes the activity "develop a migration strategy," but the model does not go into the specifics of the software development that might be required.

After studying business process reengineering methodologies practiced by leading reengineering consulting firms, Kettinger et al [1997] combined 25 of these methodologies into the BPR project stage-activity framework presented in Table 15. Once again, the model does not delve into software development, but does mention "identify IT levers" as one the activities in the "envision" stage and "implement IS" as one of the activities in the "reconstruct" stage. The authors cite a number of specific techniques used by various consulting companies during these activities. They also suggest questions that help in customizing the activities to the nature of the situation based on variables such as how radical the project is, how structured the target process is, whether the process has a customer focus, and whether the process requires high levels of IT.

As a final point about life cycles models focusing on process and organizational change, it is worthwhile to note that rich models related to the diffusion of innovation have been developed and might have been included in this article if it tried to cast an even wider net. For example, the May/August 1995 and Summer 2001 special issues of the SIGMIS publication *The Data Base for Advances in Information Systems* were devoted to the adoption, diffusion, and infusion of IT. These special issues included articles that summarized progress and directions for diffusion research (Prescott and Conger, 1995; Chin and Marcolin, 2001) along with other articles that presenting rich models of diffusion processes (e.g., Gallivan, 2001; Mathieson, Peacock, and Chin, 2001).

VII. CONCLUSION: USEFULNESS AND IMPLICATIONS OF THE WORK SYSTEM LIFE CYCLE MODEL

This article defined a general work system life cycle (WSLC) model and used that model to look at versions of over a dozen project or life cycle models

that have appeared in the IS literature. This concluding section starts with several general conclusions about the WSLC model and then discusses implications related to the communication gap between business and IT professionals and the question of which life cycle model should be used in teaching.

GENERALITY AND USEFULNESS OF THE WSLC MODEL

Many project and life cycle models can be found in the IS literature and in the literature of related fields such as general management, organizational studies, and engineering. Unless a reader is primed to think about the differences among these models, the main alternatives to any particular life cycle model and the comparative benefits of different models may be quite unclear.

A familiarity with the WSLC model might help any business professional, IT professional, instructor, or researcher identify what is or is not included in any particular life cycle model. For example, just the basic terminology of the WSLC model raises a series of questions that help in thinking about whether any other model is applicable to a particular situation:

- Iterations: Does model X contain iterations? If not, why not?
- Continuous and discontinuous change: Does model X include both continuous and discontinuous change? If not, why not?
- Development phase: Is model X based on particular assumptions about how development will be done, and does it preclude some plausible approaches to development in this situation?
- Implementation phase: Does model X include implementation in the organization? Does it address issues related to conversion to the new system? Does it address issues related to change management?
- Operation and maintenance phase: Does model X include operation and maintenance? Does it assume the system will exist indefinitely?

Does it say anything about whether and how the system might be terminated or merged into another system?

This article demonstrated the generality of the WSLC model by using it to compare over a dozen project or life cycle models. In some cases it highlighted unique concerns of a particular model. In most cases it helped in seeing how particular models emphasize specific parts of the more general life cycle and de-emphasize other parts that are also essential for sustainable systems. Most of the models were especially concerned with aspects of the development phase and much less concerned with implementation and operation and maintenance.

The useful application of the WSLC model in so many cases shows that it provides a worthwhile combination of generality and power. In contrast, most of the other models mentioned in this article do not span the other models even though they are quite useful for particular situations. For example, the software-oriented models say very little about the nature of process or organization change. Similarly, the process and organizational change models come close to assuming that the development of tools and artifacts is easy (or uninteresting) and that the real issues are mostly about topics such as analysis, design, negotiation, commitment, involvement, and power.

In addition to being useful for comparing different project and life cycle models, the WSLC might be useful to developers considering a project organized around any of those models. Embedding their initial model within the WSLC model might help them see some of the places their model should be extended or elaborated. This might be especially helpful to developers of Web sites, decision support systems, expert systems, data warehouses, and other applications that are sometimes viewed as technological marvels quite different from other types of information systems. For example, greater emphasis on creating or improving a work system could fill in gaps inherent in IT-centric models about how to build a Web site or data warehouse application. Greater attention to implementation and operation and maintenance as two of the four phases might help designers tailor these systems to realistic needs, interests, and capabilities of the users.

DOES THE USE OF SOFTWARE DEVELOPMENT MODELS INHIBIT COMMUNICATION ABOUT SYSTEMS IN ORGANIZATIONS?

There is a long history of bemoaning the gap between two cultures, be it science versus the humanities [Snow, 1959], technologists vs. non-technologists, or IT professionals versus business professionals. A decade of dealing with MBA and EMBA students who work at a wide range of companies in the San Francisco Bay Area leaves me convinced that there is often a significant communication gap between business and IT professionals. Of course most IT professionals know more about computer hardware and software, but the communication gap is about the difficulty business and IT professionals have in establishing mutual understanding that helps them communicate in both directions about their views and concerns.

Life cycle models about software development are fine for IT professionals focusing on software development, but software development simply isn't the main concern of business professionals, who care much more about how to create, improve, and sustain work systems that may or may not be supported by information systems. Consider the many essential roles that business professionals play in work system (and information system) life cycles:

- In the initiation phase, they participate in creating the project plan and the functional specification or other description of the opportunity or problem and of how work systems and information systems should be changed. (Notice that changing the information system usually addresses only part of the problem and that work system changes unrelated to the information system changes may also be needed.)
- In the development phase, they participate in specifying the detailed requirements (or in selecting and configuring vendor software), and may participate in testing the software and producing or testing documentation and training material.
- In the implementation phase, they are heavily involved in explaining the need for the change, performing and receiving training, converting

to the new information system and work system, troubleshooting, and acceptance testing.

- In the operation and maintenance phase, they perform the business operations, monitor the work system and information system, and participate in changes that are needed to sustain both systems. [Alter, 2002, Table 12.3]

With all of these roles for business professionals across an information system and/or work system life cycle, it is clear that a software development life cycle model is not an appropriate basis for mutual understanding between business and IT professionals.

Single-minded emphasis on a software development life cycle may also have negative consequences for IT professionals. Regardless of how well a particular life cycle helps organize, track, and measure the software development work, over-emphasis on the software per se highlights the IT goal of "better software within budget and on schedule" rather than the business goal of "better work systems attaining business objectives within budget and on schedule." This would not be a problem if IT professionals worked in isolation and were totally driven by the goal of producing error free software on time and within budget. However, they often need to communicate with business professionals, and for this they need an inclusive frame of reference that encompasses broader concerns about the work system (or information system) life cycle.

The main implication is not that IT professionals should use an organizational change model to develop high quality software. Rather, it is that business and IT professionals should be aware of different life cycle models that address different issues. Awareness of project and software development life cycle models is essential not only for IT professionals but also for business professionals trying to understand how required software is being developed, why that work takes so long, why and how they or their representatives need to participate, and what will happen when the real world requirements change as the business situation changes. Similarly, IT professionals need work system life cycle models both to appreciate the realities that software development efforts

address and to communicate effectively with business professionals who live those realities.

Accordingly, a range of different life cycle models should be included in the curricula for both business and computer science degrees. These curricula should include models related to software development and configuration of vendor software. In addition, the curricula should convey the difference between software, information systems, and work system and should explain why life cycle issues at the various levels are different. Business students need this range of models to understand the operational realities hidden behind the high hopes for the "digital age" and the "new economy." Computer science students need it to learn how to communicate effectively with their potential customers. Attention to a number of different life cycle models should make it 100% clear to both business and computer science students that even software sold as a "solution" doesn't solve anything until it is implemented and becomes part of a sustainable work system.

Editor's Note: This article was received on July 15, 2001. It was with the author for two weeks for one revision and was published on October 25,2001.

REFERENCES

EDITOR'S NOTE: The following reference list contains the address of World Wide Web pages. Readers who have the ability to access the Web directly from their word processor or are reading the paper on the Web, can gain direct access to these references. Readers are warned, however, that

1. these links existed as of the date of publication but are not guaranteed to be working thereafter.
2. the contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced.
3. the authors of the Web pages, not CAIS, are responsible for the accuracy of their content.
4. the author of this article, not CAIS, is responsible for the accuracy of the URL and version information.

Alter, S. (1999) "A General, Yet Useful Theory of Information Systems," *Communications of the AIS*, 1(13), March 1999. http://cais.isworld.org./articles/1-13/

Alter, S. (2001) “Are the Fundamental Concepts of Information Systems Mostly about Work Systems?” *Communications of the AIS*, 5(11), April 2001. http://cais.isworld.org./articles/5-11/

Alter, S. (2002). *Information Systems: Foundation of E-Business*, 4th ed., Upper Saddle River, NJ: Prentice-Hall, 2002.

Alter, S., P. Ein-Dor, M. L. Markus, J. Scott, and I. Vessey. (2001) “Debate: Does the Trend toward E-Business Call for Changes in the Fundamental Concepts of Information Systems?” *Communications of the AIS*, 5(10), April 2001.

Baskerville, R., J. Stage, J. I. DeGross, eds., *Organizational and Social Perspectives on Information Technology*, Proceedings of IFIP TC8 WG.8.2 International Working Conference on the Social and Organizational Perspective on Research and Practice in Information Technology, Aalberg, Denmark, June 9-11, 2000, Kluwer

Bennis, W. (2000) *Managing the Dream: Reflections on Leadership and Change*, Cambridge, MA: Perseus Publishing.

Boehm, B. (1981) *Software Engineering Economics*, Englewood Cliffs, NJ:.Prentice Hall

Boehm, B. (1988) “A Spiral Model of Software Development and Enhancement, *IEEE Transactions on Software Engineering* 21(2), May 1988, pp. 61-72.

Brown, C.V. and I. Vessey. (2001) “NIBCO’s ‘Big Bang’,” *Communications of the AIS*, 5(1), January 2001. http://cais.isworld.org./articles/5-10/

Chin, W. W. and B. L. Marcolin. “The Future Direction of Research,” *The Data Base for Advances in Information Systems*, 32(3), Summer 2001, pp. 8-12.

Cox, S., R. Dulfer, D. Han, U. Ruiz. (2001) "Data Network Life Cycles in an Engineering Consulting Firm," group paper submitted in the Professional MBA program at the University of San Francisco.

Davenport, T.H. (1993) *Process Innovation: Reengineering Work through Information Technology*, Boston, MA: Harvard Business School Press.

Davis, G.B., J.T. Gorgone, J. D. Couger, D. L. Feinstein, and H.E. Longnecker, Jr. (1997) IS *'97 Model Curriculum Guidelines for Undergraduate Degree Programs in Information Systems*. Joint publication of the Association for Computing Machinery (ACM), Association for Information Systems (AIS), and Association for Information Technology Professionals (AITP), http://webfoot.csom.umn.edu/faculty/gdavis/curcomre.pdf

Davis, L.E. and J.C. Taylor eds. (1979) *Design of Jobs*, 2nd ed., Santa Monica, CA: Goodyear Publishing Company.

Doane, M. (1998) *The SAP Blue Book*, Doane Associates Press.

Fichman, R.G. and S.A. Moses. (1999) "An Incremental Process for Software Implementation," *Sloan Management Review*, 40(2), Winter 1999, pp. 39-52.

Gallivan, M. J. "Organizational Adoption and Assimilation of Complex Technological Innovations: Development and Application of a New Framework," *The Data Base for Advances in Information Systems*, 32(3), Summer 2001, pp. 51-85.

Harrington, H.J. (1991) *Business Process Improvement: The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness*, New York: McGraw-Hill.

Hirschheim, R. and H.K. Klein. (1994) “Realizing Emancipatory Principles for Information Systems Development: The Case for ETHICS,” *MIS Quarterly*, 18(1), March 1994, pp. 83-109.

Jurison, J. (1999) “Software Project Management: A Manager’s View, ” *Communications of AIS*, 2(17), September 1999.
http://cais.isworld.org./articles/2-17/

Kettinger, W.J., Teng, J.T.C. and Guha, S. (1997) Business Process Change: A Study of Methodologies, Techniques, and Tools, *MIS Quarterly*, (21)1, March 1997, pp. 55-80.

Land, F. “Evaluation in a Socio-Technical Context,” Proceedings of IFIP W.G.8.2 Working Conference 2000, *IS2000:The Social and Organizational Perspective on Research and Practice in Information Systems,* Aalberg, Denmark, June 2000.

Lewin, K. (1951) *Field Theory in Social Science*. New York: Harper and Row, 1951.

Mathieson, K. Peacock, E. and Chin, W. W. “Extending the Technology Acceptance Model: The Influence of Perceived User Resources” *The Data Base for Advances in Information Systems*, 32(3), Summer 2001, pp. 86-112.

MacCormack, A. (2001) “Product-Development Practice that Work: How Internet Companies Build Software,” *Sloan Management Review*, 42(2), Winter 2001, pp. 75-84.

Mumford, E. and M. Weir. (1979) *Computer systems in work design – the ETHICS method*, New York: John Wiley & Sons.

Mumford, E. (2000) “Socio-technical Design: An Unfulfilled Promise?” pp. 33-46 in Baskerville, Stage, and DeGross, 2000

Nadler, D.A. (1998) *Champions of Change: How CEOs and Their Companies Are Mastering the Skills of Radical Change*, San Francisco: Jossey-Bass.

Orlikowski, W.J. and S.R. Barley. (2001) “Technology and Institutions: What Can Research on Information Technology and Research on Organizations Learn from Each Other?” *MIS Quarterly*, 25(2), June 2001, pp. 145-165.

Orlikowski, W.J. and J.D. Hofman. (1997) “An Improvisational Model for Change Management: The Case of Groupware Technologies,” *Sloan Management Review*, 38(2), Winter 1997, pp. 11-21.

Prescott, M.B. and Conger, S. A. “Information Technology Innovations: A Classification by IT Locus of Impact and Research Approach,” *The Data Base for Advances in Information Systems*, 26(2 & 3), May/ Aug 1995, pp. 20-41.

Sherrell, L.B. and L. Chen. (2001) “The W Life Cycle Model and Associated Methodology for Corporate Web Site Development,” *Communications of AIS*, 5(7), April 2001. http://cais.isworld.org./articles/5-7/

Schein, E.H. (1987) *Process Consultation: Lessons for Managers and Consultants*, Volume II, Reading, MA: Addison-Wesley.

Snow, C.P. (1959) *The Two Cultures and the Scientific Revolution*, Cambridge, UK: Cambridge University Press.

Strassmann, P.A. (1997) *The Squandered Computer: Evaluating the Business Alignment of Information Technologies*, New Canaan, CT: The Information Economics Press.

Truex, D.P., Baskerville, R. and Klein, H. “Growing Systems in Emergent Organizations,” *Communications of the ACM*, 42(8), August 1999, pp. 117-123.

ABOUT THE AUTHOR

Steven Alter is Professor of Information Systems at the University of San Francisco. He holds a B.S. in mathematics and Ph.D. in management science from MIT. He extended his 1975 Ph.D. thesis into one of the first books on decision support systems. After teaching at the University of Southern California he served for eight years as co-founder and Vice President of Consilium, a manufacturing software firm that went public in 1989 and was acquired by Applied Materials in 1998. His many roles at Consilium included starting departments for customer service, training, documentation, technical support, and product management. Upon returning to academia, he wrote an information systems textbook whose fourth edition was published in August 2001 with a new title, *Information Systems: Foundation of E-business*. His articles have appeared in *Harvard Business Review, Sloan Management Review, MIS Quarterly, Interfaces, Communications of the ACM, Communications of the AIS, Futures, The Futurist,* and many conference transactions.

Communications of the Association for Information Systems

ISSN: 1529-3181