Journal of Computer Information Systems

ISSN: 0887-4417 (Print) 2380-2057 (Online) Journal homepage: http://www.tandfonline.com/loi/ucis20

A System Development Methodology for ERP Systems

Niv Ahituv, Seev Neumann & Moshe Zviran

To cite this article: Niv Ahituv, Seev Neumann & Moshe Zviran (2002) A System Development Methodology for ERP Systems, Journal of Computer Information Systems, 42:3, 56-67

To link to this article: http://dx.doi.org/10.1080/08874417.2002.11647504

A SYSTEM DEVELOPMENT METHODOLOGY FOR ERP SYSTEMS

NIV AHITUV, SEEV NEUMANN, and MOSHE ZVIRAN
Tel-Aviv University
Tel Aviv 69978, Israel

ABSTRACT

An ERP (Enterprise Resource Planning) project is considered highly risky, since it is large, complex, usually unfamiliar to the organization and implemented under a tight timetable. It usually entails process reengineering and many changes. To reduce the risk and improve the probability of project success, an organization can use a structured development approach for such a project, beginning with the selection stage and culminating in the operation stage.

There are several conventional structured development approaches. The major ones are the ISDLC model, the prototyping approach and the software package life cycle model. The implementation of an ERP system requires a new methodology that combines components from each of the above approaches. The model suggested in this paper is comprised of four stages: (1) selection; (2) definition; (3) implementation; and (4) operation. Several organizations that have adopted this structured methodology have evidenced a successful ERP implementation

INTRODUCTION

An ERP (Enterprise Resource Planning) system is an integrative information system that supports the work processes and resource management of an organization. The system is comprised of several applications whose integration supports the management of the organization's supply chain and its ongoing operation.

An ERP implementation project is usually very large and complex and combines an implementation of a new information system (IS) together with a redesign of work processes (7, 29). In most cases, this is the largest IS project ever implemented in the organization. The project is highly risky because it involves complexity, large size, high costs, a large development team and usually a tight time schedule (10).

In an ERP implementation, there must be a mutual fit of the system to the organization and of the organization to the system. A study that examined 310 manufacturing organizations in Israel found that different manufacturing organizations gain different benefits from their similar IS applications related to ERP, as a function of their organizational characteristics (23). In addition, most implementations included additions and enhancements to the system in order to support functions and tasks that were unique to the specific organizations.

An organization that can use a structured and well-defined methodology for implementing an ERP system, progressing from the selection stage to the operation stage, can reduce the degree of risk and improve the probability of project success. It is claimed that an organization that adopts a structured approach to an ERP implementation can realize 10% savings in total IT costs over five years (10).

The literature suggests various models for developing an IS application; this paper reviews three main methodologies:

- The conventional information system development (waterfall) approach that deals with developing an application;
- The prototyping approach that focuses on using a prototype during the development of an application;
- The life cycle model of an application software package acquisition and implementation that describes the process of organizing and implementing an application package.

The literature does not suggest, however, any model that describes a structured development and implementation process for an ERP package. The conventional models do not cope well with the need for the mutual fit between the ERP package and the organization. The waterfall and prototyping approaches fit the developing of an application customized to the organization; the application software package approach fits the selection and implementation of an "as is" application. An ERP system requires the characteristics of both approaches.

The purpose of this paper is, therefore, to design a new development model that will combine main traits from all traditional approaches and be compatible with the unique traits of an ERP project. The model will be generic and could thus be useful for implementing an vendor's ERP package, in any industry and in any market (domestic or international).

TRADITIONAL SYSTEM DEVELOPMENT MODELS

This section reviews three conventional approaches to developing an information system application.

Information System Life Cycle (The Waterfall Model)

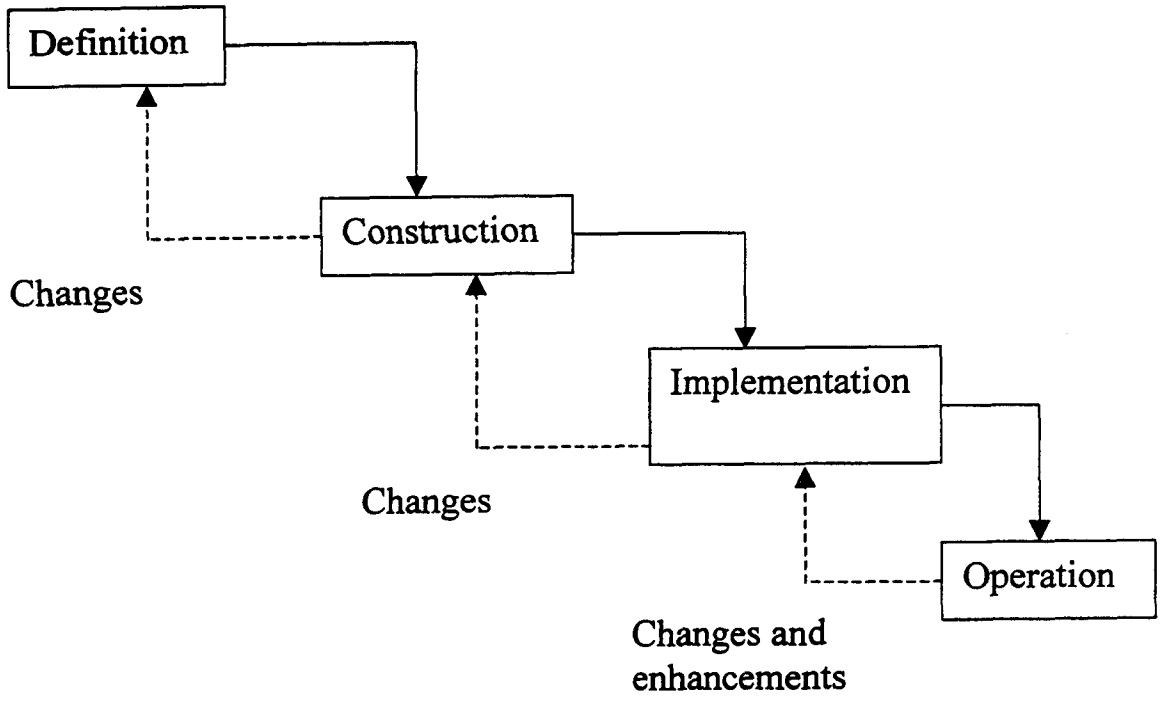
The phases of development through which any individual IS application goes range from an initial information requirement – problem or opportunity – through the design, construction and operation of the system, to the eventual termination of the system, which triggers a life cycle of a new system (28, 30). The development of every information system application is a project-oriented undertaking that follows a uniform pattern consisting of a structured series of activities. Every one of these major activities can be broken down into smaller task; each of these tasks almost invariably occurs in every application development project.

The *life cycle* of a computerized information system contains the following phases and activities (Figure 1):

- Definition phase (preliminary analysis, feasibility study, information analysis, system design)
- Construction phase (programming, procedure

development)
- Implementation phase (conversion)
- Operation phase (operation and maintenance, postaudit, termination)

FIGURE 1
Information System Development Life Cycle Phases



A *preliminary analysis* is usually conducted before an information system is designed and implemented. The analysis is to determine if the problem or the subject due for development warrants further analysis. It also refines the problem statement and creates a preliminary plan for an in-depth analysis of the problem.

The major purpose of the *feasibility study* is to establish whether a project should be done and how it should be done if justified. The major purpose of this activity is to determine whether it is feasible to develop and install a system.

The *information analysis* activity is sometimes justifiably called the *logical* (or *conceptual*) *design step*, to distinguish it from the next activity of *system design*, sometimes called *physical* or *technical* design. Because of the relatively large scope of this activity, it usually consists of three parts: (1) analysis of present system, with the emphasis on *what* is taking place in the system; (2) determination of the information requirements, with the emphasis on *why* they cannot be provided effectively or efficiently by the current system; and (3) conceptual design of the new system.

System design is the creative activity of devising the program and procedure specifications for processing data by the new system. It involves the development of complete and detailed programming specifications from which the programmers can proceed with little or no additional outside reference.

During the *programming* phase, coding and testing of computer *programs* take place. The programs are then tested by actual execution on the computer system.

While the programming activity results in computer instructions, the *procedure development activity* results in instruction for human involvement with the new system. Procedures for the various users and operators of the system are written and tested. This activity concurrently proceeds with the programming activity and is equally important.

Conversion refers to (1) *training* the personnel operating and using the new system, (2) breaking the system in, and (3) acceptance testing by the user. In most cases conversion is associated with making changes from an old system to a new system.

At the conclusion of the conversion activity, the system moves into the operation phase of life. The development activities of the information system life cycle are now terminated. The system now operates like a production facility, processing data, producing information and undergoing maintenance.

Periodic postaudit review form control points throughout the operation of the system. These reviews will indicate when the life of the current system is drawing to a close and a new life cycle is indicated. *Termination* is imperative when requests for changes and the number of errors reach a level where the continued operation of the current system is not worthwhile.

The structured life cycle of an information system may convey an impression of a completely linear process, where each step must terminate before the next one starts. However, not all activities are necessarily performed linearly. Some may overlap or be performed in a parallel manner. Others cannot be started before the conclusion of a preceding activity. The life cycle thus

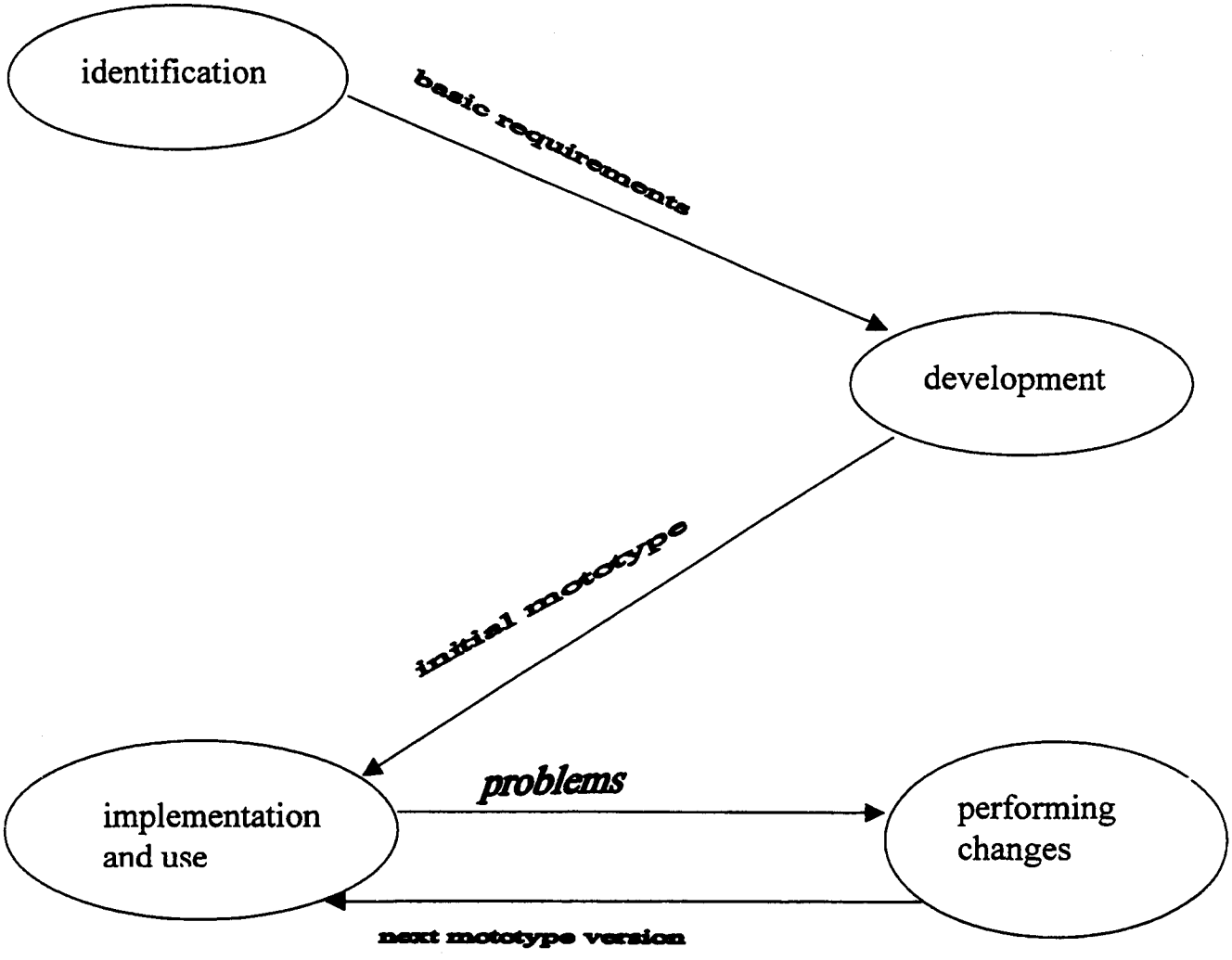
changes from organization to organization, from system to system, and from period to period. The only aspect that is completely linear is the succession of management checkpoints throughout the life cycle.

The Prototyping Model

The prototyping approach to systems development is, in many ways, the very opposite of an old-style SDLC. Instead of spending a lot of time producing very detailed specifications, the developers find out only generally what the users want. The developers do not develop the complete system all at once. Instead they *quickly* create a prototype, which either contains portions of the system of most interest to the users or is a small-scale working model of the entire system. After reviewing the prototype with the users, the developers refine and extend it. This process continues through several iterations until either the users approve the design or it becomes apparent that the proposed system cannot meet their needs. If the system is viable, the developers create a full-scale version that includes additional features.

Figure 2 displays a four-phase model for developing a prototype (3, 20).

FIGURE 2
System Life Cycle Based on a Prototype



The *identification* phase defines the main user requirements. Because the requirements are not well understood, users and developers start by building a prototype instead of writing a functional specification.

The *development* process emphasizes speed and rapid feedback. It begins by developing an initial prototype that demonstrates some of the desired processing but is far from complete. Next come a series of iterations, each modifying the prototype based on user comments about the most recent version. Once the requirements are clear, the users and technical staff must decide how to proceed. They might conclude that the project should be abandoned. If the project is to continue, one approach is to complete the prototype using the code that has been generated thus far. Another is to shift to a traditional

system life cycle by writing a specification based on what has been learned and then doing the design based on the desired level of information system performance. If the system is primarily for management reporting or if business problems are changing rapidly, extending the prototype might be a better choice.

When using a prototype, part of the implementation is done in parallel with development. Users try out the prototype during successive iterations and become familiar with what it does and how it can help them. Systems developed this way may require less user training in the implementation phase. The conversion step should be similar to that of the traditional approach. In the last phase of the prototyping approach, *changes* required by the previous phase are used to develop a new version of the prototype. The last two phases are iterative, until the prototype is accepted by the users.

At the end of the four phases and after the user has accepted the prototype, the development of the real information system application can start. There are three methods for switching from the prototype to a full-blown system (2, 9, 15).

The Application Software Package Model

Purchasing an application package reduces the time delay until a system can be operational. It also reduces the amount of system development work that is needed. However, as will be apparent by looking at the phases of acquiring and using an application package, the life cycle for those systems still requires a great deal of effort. It might seem that buying an application system from a vendor would bypass most of the work of building and maintaining the information system. In fact, the company's staff must still work on all phases to ensure that the right application package is selected and is set up and supported properly.

The major advantages of application software package implementation are lower costs and faster implementation (18). The cost of implementing an application package is usually lower than that of customized developments because the vendor can spread the development cost over many clients. Also, the time required to implement an application software package is shorter than that of customized developments because coding, testing, and debugging are unnecessary. However, the need for modifications to an application software package reduces these advantages (13). But the need for such modifications can be reduced if systematic managerial efforts are devoted to analyzing requirements, evaluating candidate packages, and implementing the acquired application software package.

Shin and Lee (25) suggest an application software package life cycle model consisting of three phases that are divided into 25 activities that are performed sequentially or in parallel.

The first phase, *project formulation* consists of eight activities. The objective of the phase is to capture users' needs in order to define the criteria for comparison among application software package vendors. The phase can be divided into two sub-phases:

- Project initiation – this sub-phase consists of four activities performed sequentially:
  - Appointment of a project committee.
  - Definition of problems to solve.
  - Definition of scope and goals of the project.
  - Planning the overall selection and implementation process.
- Analysis of needs – this sub-phase consists of four activities performed sequentially:
  - Analysis of the current system.
  - Definition of users' requirements.
  - Investigation of organizational constraints.
  - Integrating the results of the preceding seven activities.

The second phase, *application software package selection and acquisition*, consists of nine activities. The phase can be divided into three sub-phases:

- Preparation – this sub-phase consists of four activities performed sequentially.
  - Searching for information about candidate packages.
  - Evaluating and listing alternative packages.
  - Developing an RFP.
  - Developing a model for evaluating the packages suggested by vendors responding to the RFP.
- Selection – this sub-phase consists of four activities performed sequentially:
  - Analysis of vendors' proposals.
  - Conducting interviews with vendors.
  - Testing the proposals and/or conducting site visits.
  - Selecting the best system and getting management approval.
- Acquisition – this sub-phase consists of one activity: contractual negotiation with the vendor, issuing the purchase order and receiving the application software package.

The third phase, *installation, implementation and operation*, consists of eight activities. The phase can be divided into two sub-phases:

- Installation and operation – this sub-phases consists of seven activities:
  - Developing the implementation plan.
  - Modifying the application software package, if needed.
  - Modifying organizational processes to fit the application software package, if needed.
  - Training of users.
  - Data conversion.
  - Acceptance of the package in the users' environment.
  - Operation and maintenance of the package.
- Performance evaluation – this sub-phase consists of one activity: periodic evaluation and audit of the application software package performance and, eventually, a decision to terminate the use of the package and to start a new life cycle process.

CHARACTERISTICS OF AN ERP DEVELOPMENT METHODOLOGY

Section 2 presented three structured development models. In many aspects, an ERP system is an extension of an MRP system, which is acquired as an application software package or developed in-house. An ERP life cycle approach could therefore be based on one of these models. However, because of the ERP life cycle characteristics discussed in this section, it is claimed that the ERP life cycle approach has to combine features from those three models.

Several characteristics affect the ERP life cycle, its objectives and its phases. Figure 3 exhibits nine main characteristics that shape the ERP life cycle.

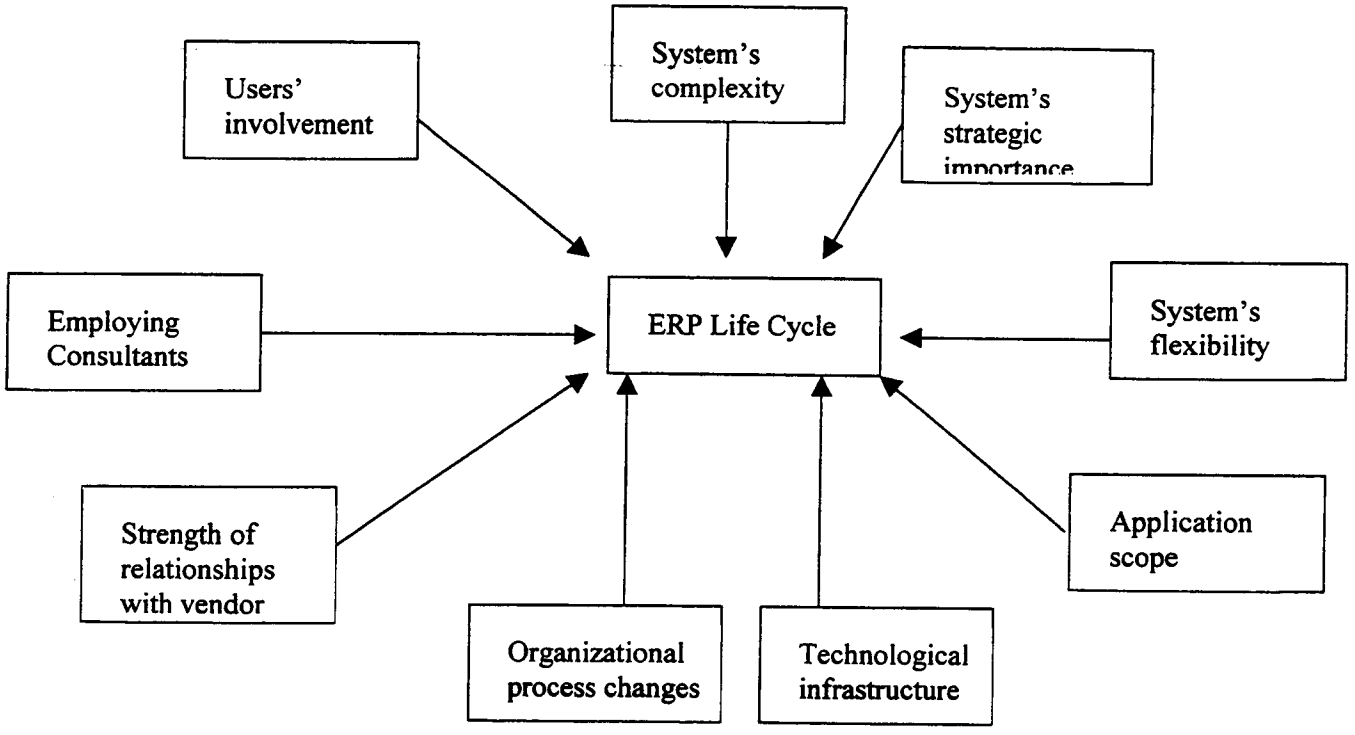
Other characteristics that are not unique to an ERP system (such as "type of organization") similarly affect the development of any IS application but are not included here.

Complexity

The simplest system includes one application that can be operated independently of other organizational applications or that has a very friendly user interface. A highly complex system

includes many applications that are internally linked and also are linked to other systems in the organization. Usually, the difficulty in such systems is in information integrity and in managing the interfaces (linkages).

FIGURE 3
Characteristics Affecting the ERP Life Cycle



An ERP system is the most complex system today within the realm of information systems. It includes many applications that support many organizational processes. The degree of differentiation among the applications is high and the degree of difficulty in implementing and maintaining each one is different. Complexity engenders high risk (19). Because of its complexity, determining the system's boundaries is one of the most important subjects in the beginning of the project. In order to achieve implementation success, the project has to be bounded and the ERP applications that will be implemented (and when) must be clearly defined in the beginning of the project.

Strategic Importance

A strategic (competitive) IS application is one that has an impact on the organization's success and that supports its strategy (21). Such an application can pose a great risk to the organization. A non-strategic application supports organizational operations that are not crucial to its success; hence, it does not pose a great risk. An ERP system is highly strategic since it supports all organizational processes. At the same time, the system includes a significant number of non-strategic applications (e.g., a human resources module).

Because of the crucial role and the high risk of an ERP system, it is important to select the best system and a reliable vendor who will support the system for many years. Many organizations also tend to appoint the CEO or a senior VP as the ERP project leader and to manage the project rather than to outsource it because of the strong linkage of the project to corporate strategy and of their familiarity with the organizational processes (27).

Flexibility

An ERP system is an application software package that excels in its flexibility compared to other packages. Flexibility can be described by four dimensions:

- *Parametric flexibility* – the system can be designed according to the organization's needs by defining its parameters.
- *Code changes* – system segments can be changed by changing the relevant software segments. The changes can be performed by the vendor, a consultant, or by the IS department.
- *Module addition* – the system enables the addition of software segments that constitute additional modules to the system. The additions can be implemented, again, by vendors, consultants, or by the IS department.
- *Connectivity to other systems* – satellite systems can be linked to the ERP system in order to complement data processing and analysis (for example, a data warehousing application).

The flexibility is particularly important to the system's strategic importance because the organization is interested (as much as possible) in a unique customization of the system in order to reap competitive advantages.

Application Scope

The wide scope of an ERP project, which aims to provide one overall solution and its inter-functional nature cause unique problems that require management's involvement during implementation (6, 7). The problems are:

- Project control.
- Need for integration among organizational units.
- Difficulty in determining timetables.
- Increased project risk.
- High implementation cost.

Technological Infrastructure

An ERP system is based on advanced technologies. In most organizations, the move to the new system requires the replacement of the existing infrastructure. This activity increases the risk level of the project since it involves an additional capital investment, appropriate personnel skills and even the possibility of a temporary shutdown of business. For this reason, the activities relevant to infrastructure changes (e.g., identification of new technology) must be incorporated in the ERP life cycle.

Organizational Process Changes

An ERP project includes a massive change of work processes and information flows. Naturally, introducing changes is a difficult political process that can trigger resistance from conservative organizational bodies and persons. There is, therefore, a need to add marketing elements when selling the system to the various departments (5) and to involve management in decision-making (8). Also, the selection of the most appropriate ERP system impacts the number of organizational changes (17, 18).

Another ERP life cycle implication is that there is no need for a detailed RFP, since ERP systems cater to the standard processes in organizations in the relevant industry. An RFP for an ERP system should, therefore, cover only the processes unique to the organization.

Intensity of Relationship with Vendor

The intensity of relationship changes according to a system's importance to the organization – the higher the importance, the more intense the relationship. Since an ERP system is highly strategic to the organization, is highly complex and has a wide scope, the relationship with the ERP vendor is quite intense. It is clear that project success depends on a joint effort of the vendor and the organization.

During system implementation, the intensity depends on the vendor's experience in similar organizations. The relationship with the vendor continues into the phase of operation and maintenance, when changes are performed and the system may be upgraded. Relying on a single vendor increases the degree of project risk (26). Therefore, during the selection phase, the vendor's financial conditions must be heavily weighed.

Employing External Consultants

Employing consultants in any application software package implementation depends on several factors: system's scope, the experience and know-how of the IS unit in implementing similar systems and, of course, the consultant's experience in similar projects. An ERP project requires many skilled human resources during implementation who are not needed later during operations. Many organizations prefer, therefore, to employ, during implementation, experts from consulting companies that specialize in ERP projects.

Users' Involvement

Research shows that the interaction between users and developers is very important (12, 16) and that the degree of users' knowledge affects project success (19). Some researchers argue that users must be the driving force of the project (4).

The degree of users' involvement in an application development depends on their past experience in similar projects (24). An experienced user can make decisions on project issues related to his or her function. An inexperienced user should delegate to the developers more decision-making responsibility.

AN ERP DEVELOPMENT MODEL

This section suggests and describes a generic model for an ERP system development life cycle. The definition of life cycle phases is based on the following sources:

- Existing life cycle models described in Section 2.
- System implementation methodologies developed by systems vendors.
- A literature search on various subjects pertaining to information systems implementation.
- Case studies identified during the literature search.

The ERP life cycle model is described in Figure 4.

During the *selection phase*, the standard activities and those unique to the organization are reviewed in order to determine the project boundaries; system's vendor and external consultants are selected.

During the *definition phase*, the system components that will be included in the implementation are defined and the implementation plan is prepared.

The *implementation phase* is the core of the life cycle model. During this phase, the organizational processes are redesigned to work with the ERP system, the system is implemented, user training and acceptance testing is performed. At the end of this phase, either changes to the system are performed or the project progresses to the operation phase. This phase is iterative – the number of component/process implementations is defined in the definition phase. Each iteration begins at the conclusion of the acceptance tests of the previous iteration.

During the *operation phase*, the ERP system is in operation, is maintained and is upgraded, if needed.

The following is a breakdown of these four major phases into more detailed steps.

The Selection Phase

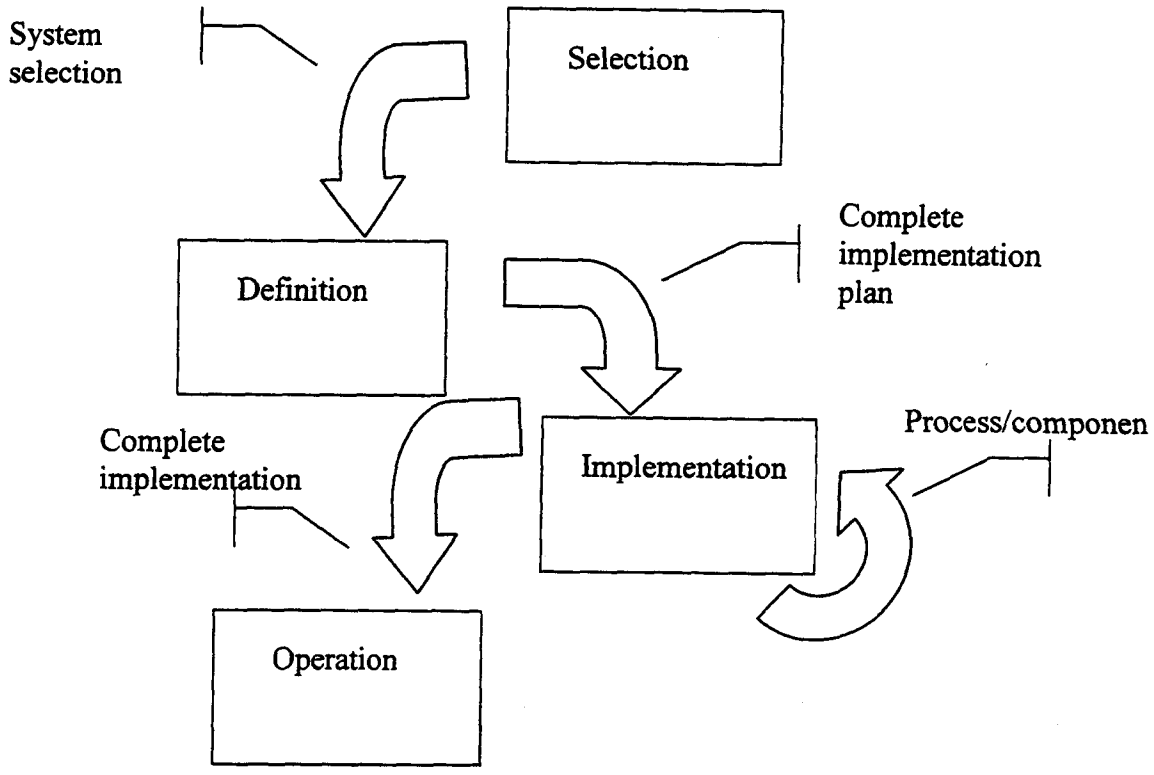
The objective of this phase is to identify the ERP package most appropriate for the organization and the technological infrastructure needed for it. When the decision is to implement components from several ERP packages, the objective is to identify the optimal components. If additional external human resources are needed, the selection of the consulting (or outsourcer) firm is also part of this phase.

The phase is comprised of nine activities:

1. *Definition of project objectives* – analysis of the reasons for an ERP system, identification of gaps and weaknesses in the current information systems and definition of objectives for the new system. Within this definition, the project

boundaries are defined and a decision is made whether to use consultants and at what project steps. The objectives are defined by an ERP project steering committee that is appointed for this project.

FIGURE 4
The ERP Life Cycle Model



2. *Collection of information about systems and vendors* – an RFI is usually issued to prospective vendors of ERP systems. Additional information is collected from colleagues, IT industry information firms (e.g., The Gartner Group, GIGA), professional fairs and conferences, professional literature and the like. At the end of this step, the information is presented to the steering committee. ERP systems that do not meet the objectives defined in the previous step are ruled out.
3. *Collection of information about consulting firms* – this step is taken if the steering committee has decided to use consultants. An RFI is sent to prospective firms (or individuals); additional information can be collected from colleagues, ERP system vendors, and IT industry information firms. At the end of this step, the steering committee decides whether to continue with the project; if so, about three to five vendors and consultants are selected for more profound investigation in later steps.
4. *Needs analysis* – analysis of organizational needs. All processes, computerized or manual, are analyzed, including the feasibility and need for new processes. The analysis is done by joint teams of IS professionals and key users. At the end of this step, if approved by the steering committee and corporate management, an RFP booklet is produced. The booklet is sent to the vendors and the consultants.
5. *Investigation of vendor alternatives* – after receiving responses to the RFP, meetings are held with vendors. Vendors can present applications and the advice of vendors' existing customers can be solicited. At the end of this step, information about the vendors is kept for final analysis during the feasibility study step (Step 8).
6. *Investigation of consultant alternatives* – in parallel to the previous step, responses to the RFP are evaluated and meetings are held. The information is then presented to the steering committee.
7. *Collection of information on the technological infrastructure* – this activity is performed by the IS department, after receiving the vendors' responses to the RFP. An RFI is sent to hardware and telecommunications vendors, soliciting possible technological solutions compatible with the ERP vendors' responses to the RFP sent to them. At the end of this step, the information is kept for future analysis during the feasibility step.
8. *Feasibility study* – after collecting all the information about ERP and technology vendors and consultants, IS professionals and key users perform a feasibility analysis of the various alternatives. The analysis focuses on three aspects – technological, economical and organizational feasibility.

   At the conclusion of this step, a feasibility report is submitted to the steering committee and senior management for approval. The report includes recommended alternatives and the rationale for continuing the project, reevaluating it, or killing it. If the decision is to continue, the ERP system, the technological infrastructure and the consulting firm are selected.

9. *Contract negotiation and signing* – contracts are negotiated and signed with all of the external entities – the ERP system vendor, the infrastructure vendor and the consulting firm. The development and then the management of the contracts is a prerequisite for the project success. When components from several vendors are implemented, the contracts have to refer to the integration among the components in order to eliminate potential future frictions between the vendors and the organization. The negotiation should also cover the pricing schemes (including penalties and bonuses).

The Definition Phase

This phase is the shortest of the four ERP life cycle phases. It includes all the preparatory activities for the implementation phase that follows. The phase is comprised of three steps.

1. *Definition of project scope* – project scope (boundaries) is determined on the basis of the system selected in the previous phase. This activity is performed by personnel from the IS department and user representatives in cooperation with the vendor and the consultants. The joint team determines the implementation method and the order in which components and ERP modules will be implemented. The recommendations are approved by the steering committee.
2. *Establishing implementation teams and timetables* – the teams include personnel from the ERP vendor, from the consulting firm and from the organization. The division of responsibilities among the three depends on each one's experience in ERP implementation, in implementing IS applications in general and in managing an organizational change.

   The teams are established by the ERP project leader together with the steering committee. In addition to the ERP implementation teams, a technical team (comprised of similar personnel) is established to implement the technological infrastructure. The project leader, together with the team leaders, then determines the timetables for the implementation activities and the system rollout.
3. *Training of the implementation teams* – the objective is to train key users and management representatives, who are members of the implementation teams, in the use of the selected system. This will enable them, in the next step, to compare the current work processes with those offered by the ERP system. The training is facilitated by the system vendor.
4. *Initial implementation of the system* – in parallel to the training step above, system components are implemented in key users' sites. This implementation facilitates the construction of prototypes by the implementation teams and, hence, does not have to involve all the users. The implementation enables the users in the implementation teams to exercise the subjects covered in the training and to get acquainted with the technological infrastructure.

The Implementation Phase

This is the main phase of the ERP life cycle. Its objective is to link the ERP system to the organizational processes so that when the system moves to the operation phase, its contribution to the organization would be at the maximum.

The implementation phase is performed iteratively, according to one of the following methods:

- *Adding processes* – in each successive round, one or more additional processes are implemented in all organizational sites.
- *Adding organizational layers* – in each successive round, a new organizational entity is added to the computerized processes. This method facilitates getting a clear picture of system use in all activities at an early stage.

These iterations, on one hand, reduce the project risk by an early detection of implementation problems. On the other hand, they enable improvement of the implementation, training of teams and a reevaluation of the next steps of the life cycle. An update of the system is rolled out to the users after each iteration and the implemented process enters the operation phase.

This implementation method is unique to ERP systems. When the traditional SDLC makes use of an iterative development it does so due to new demands from the system. In ERP systems, the iterativeness is intended to reduce the implementation risk and is needed because of the strategic importance of the ERP system and its large scope.

The implementation phase comprises nine steps:

1. *Gap analysis* – the objective is to identify gaps between the definition of processes in the ERP system and their definition in the organization. The processes described in the RFP are investigated as to whether they can be implemented as described, whether complementary solutions are needed, or whether changes in the system or in the processes are necessary. The recommendations can be to change organizational processes, to add complementary processes, or to expand the ERP system. At the end of the step, a gap analysis report is prepared. The report describes the processes where gaps have been detected and recommendations of how to handle the gaps. The report is presented to the steering committee for consideration and approval.
2. *Business process reengineering* – BPR has been defined as an analysis and design of work flows and processes within and between organizations (11) in order to achieve a drastic positive change in performance (14). Performing BPR as part of an ERP project is recommended immediately following the gap analysis step. Sometimes BPR is performed in parallel with the gap analysis or even before the implementation phase.
3. *Identification of complementary solutions* – implementation of solutions complementary to the ERP system is investigated together with the BPR step. For example:
   - Development or acquisition of additional modules that are integrated into the system.
   - Development or acquisition of complementary systems that can interface to the ERP system.
   - Additional manual work processes.

   The complementary solutions can be regarded as a sub-project to be implemented according to the project's timetable. The selection of solutions is approved by the steering committee and senior management.
4. *Construction of a prototype* – the objective of the prototype is to test the system design and the integration of the new processes and the complementary systems. The prototype facilitates the testing of the new workflows and the *constraining* of users' expectations concerning the designed processes. It also reduces risks by reflecting the final model of operation in a relatively early stage.

   Like the ERP implementation phase and as typical to the prototyping model (described in Section 2 above) the construction of the prototype is iterative. In order to save time, it is recommended to build the prototype in conjunction with the continuing design of new processes; as such, it is a rolling and evolving prototype.

At the end of building the prototype and after the processes designed are approved, the prototype becomes the full-blown system and is rolled out to all the users.

5. *Data conversion* – performed in parallel to the construction of the prototype. The conversion of data from old applications can be done by manual data entry or by using conversion software.
6. *Definition of work procedures* – the objective is to update organizational procedures. The new or updated procedures include general procedures and those specific to working with the new ERP system. This activity is performed by key users. At the end of the step, a new procedure manual is produced and is approved by senior management.
7. *Full implementation of the system* – after the prototype is approved and the data have been converted, the full version of the ERP system is implemented in the relevant organizational units. This is usually done in parallel to the existing system, in order to minimize the disturbance to the ongoing operation.
8. *Training of users* – training can be done by lectures or by self-study while working with the implementation teams. Since training costs are quite high, it is recommended to prepare a training plan that will provide an overall framework and facilitate control of the activity.
9. *Acceptance tests* – the objective is to test with real life data the new ERP system and its linkage to other organizational (non-ERP) systems. The tests include the testing of the modules that were added to the system. The emphasis is on system capacity and throughput and interfaces with other systems. The tests must be conducted in a process orientation – process scenarios have to be prepared and tested with real-life data.

The Operations Phase

This phase is the longest phase of the ERP life cycle and can last several years. Five steps are included in this phase.

1. *Establishment of support centers* – these constitute a learning network in the organization, whose objective is to develop user training tools and to support ongoing operations. The need for a support center increases directly with the complexity of the system.
2. *Performance of changes and enhancements* – these are needed over time due to organizational dynamics, changes in business strategy, technological and environmental changes and the like. They can be handled by adding new processes to the system or by expanding existing ones.
3. *Upgrading the system* – the objective in this step is to watch the software updates of the vendor and the technological changes and to upgrade the ERP modules by adding activities and functions facilitated by the updates.
4. *System audit* – similar to other life cycle models, the objective is to verify whether the system meets users' needs. This audit is performed periodically.
5. *System termination* - this step is analogous to the termination activity of the traditional SDLC. Due to the strategic importance of an ERP system and its close linkage to organizational processes, it seems that replacing the system is much more complex and difficult than replacing conventional applications.

COMPARING THE ERP MODEL TO OTHER DEVELOPMENT MODELS

The ERP life cycle model, as described in the previous section, is based on the integration of the three life cycle models discussed in Section 2. The integration is needed because of the characteristics unique to the ERP system (e.g., complexity, strategic importance, scope, flexibility, intensity of relationships with vendors and consultants, etc). The four life cycle models can be compared within the framework of four major phases, as exhibited in Figure 5.

Note that in the following discussion, we present the phases mentioned in the description of the life cycle models. We are not arguing that a certain phase is not actually performed, but that a certain model does not include it, because the phase is not unique or does not significantly impact the model. By comparing the four models, two main observations emerge that demonstrate the essence of the difference between the models.

- The selection phase is not part of the SDLC model and the prototyping model, since in these models the systems are usually developed internally, or because the selection of a vendor is a marginal activity, relative to the other life cycle activities. In the life cycle model for an application software package, the system selection takes place after the definition phase, while in an ERP life cycle model it takes place in the beginning of the project.
- In the three models, the definition phase is performed before the development and implementation phase. In the ERP life cycle model, since the essence of the model is the mutual fit of the system and the organization, the definition takes place in parallel to the process of development and implementation. For this reason, in an ERP project, the time allocated to the definition activities is shorter than in the other models.

Following is a more detailed comparison of the activities in the four models.

The Selection Phase

An ERP system and an application software package, which are acquired from an external vendor, require attention to the process of selecting the system. The SDLC and prototyping models do not include a selection phase, since in most cases, the supplier is the internal IS unit and not an external vendor.

Collection of information about systems and implementations – in the ERP model, this activity is performed before the analysis of needs, due to the large scope and complexity of the system. In an application software package model, which focuses on less complex systems, this activity is performed after the analysis of needs.

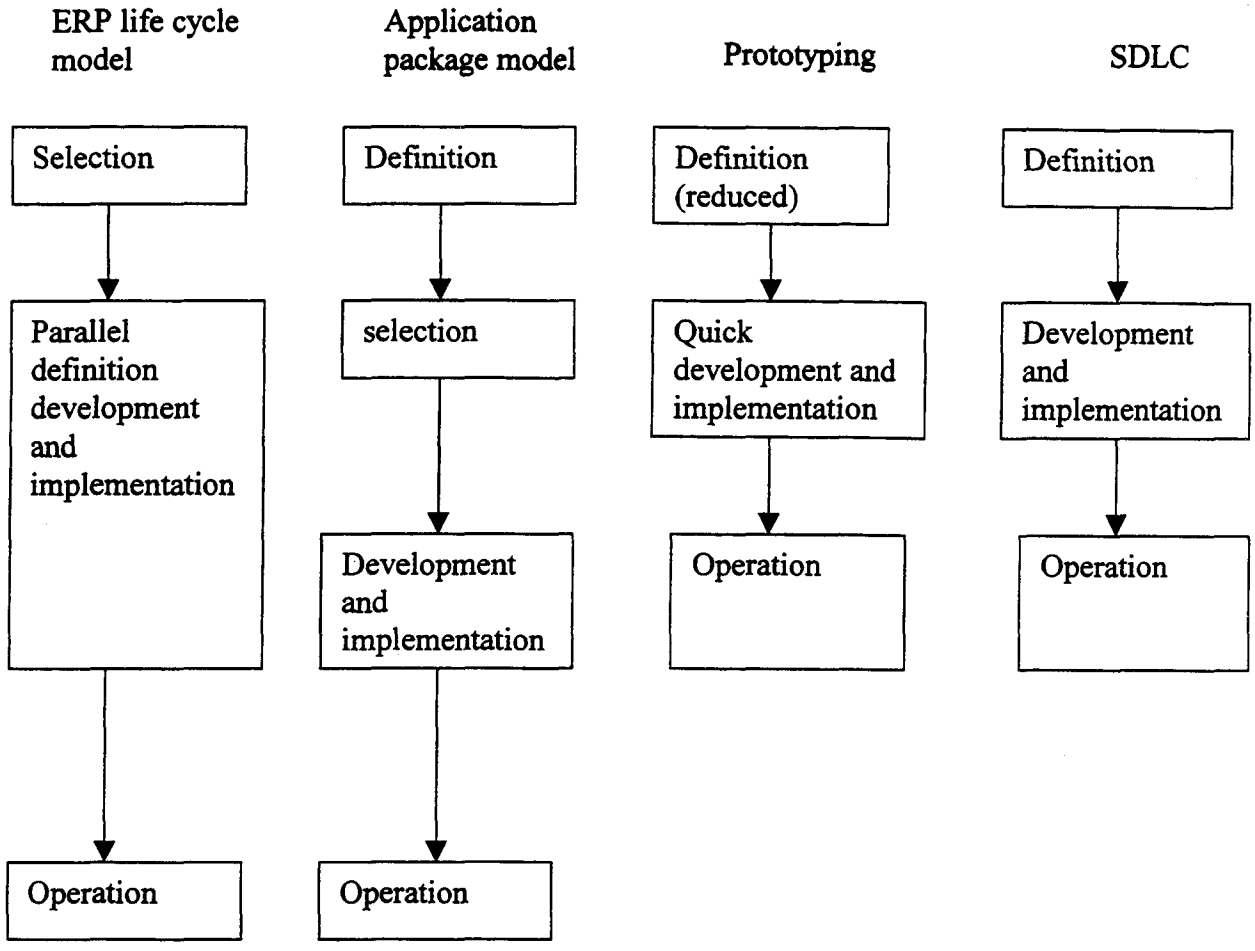
Needs analysis – this activity is similar to the preliminary analysis step of the SDLC model. However, while the preliminary analysis is a basis for the information analysis step, the needs analysis is the basis for writing the RFP.

In the ERP life cycle, the RFP includes only processes unique to the organization and not standardized (generic) processes, since all ERP systems provide similar solutions to the latter. In the other models, the RFP refers to all the processes in the organization and takes, therefore, a long time to prepare.

Collection of information about the technological infrastructure – in an ERP project, the selection of a certain system triggers in many cases a change in the infrastructure; hence, the model has to address this activity. The selection of technology in the SDLC and application software package models usually depends on the organization's IS strategy; hence, there is no attention in these models to this activity.

Contract signing – the SDLC model does not include this activity. The application software package model includes it, although the contractual process is relatively simple.

FIGURE 5
Phases of the Life Cycle Models



The Definition Phase

While the definition phase is part of all four models, there is a significant difference between the activities comprising this phase in the various models.

Definition of project scope – in the SDLC model this activity is performed as part of the consideration whether to develop a new system.

Establishing implementation teams – in an ERP project the implementation teams are responsible for the whole implementation process, from system definition to testing. The teams include internal personnel, consultants and vendor personnel. Judicious staffing of the teams and their integration are vital to project success.

In other models, the systems being developed are usually internal, do not include external personnel and, therefore, do not require similar integration.

Training of implementation teams – in the ERP life cycle, training is done before the implementation phase in order to provide knowledge about the system before performing the gap analysis activity. In the SDLC model, training is delivered after the coding of the software. In the application software package model, training is delivered after system design.

Initial implementation of the system – an initial implementation of an ERP system is performed during the definition phase only for the implementation teams, to facilitate the gap analysis activity. Later on the system is implemented for all users. In an application software package life cycle, the system can be implemented quite early for all users. In a SDLC model, the system is implemented only after its development and testing.

The Implementation Phase

The implementation phase of the ERP model can be compared with the construction and implementation phases of the SDLC model. This is because the construction of the ERP system takes place in the implementation phase (when the system prototype is built), while in the SDLC model it takes place in the construction phase.

Similarly, the ERP implementation phase should be compared with the two phases of development and implementation of the prototyping model. In the application software package model, the implementation phase is united with the operation phase.

The iterativeness of the implementation phase is a basic

underlying principle of the ERP model due to the wide scope, complexity and strategic importance of the system. In the SDLC model, iterativeness is less of a principle and, in simple systems, does not even exist. In the application software package model, iterativeness is not addressed, probably due to the relatively narrow scope of these applications.

Gap analysis – gap analysis is performed in order to investigate the system's fit with users' requirements. In the SDLC model, the system is developed according to users' requirements and there is no need for gap analysis. The ERP and the application software package application software package life cycles do include this activity.

Business process reengineering – an ERP project combines this activity with the development of an information system. In an application software package model, most of the changes relate to the software and not to organizational processes. In a SDLC model, the system is developed on the basis of users' requirements and this activity, therefore, is not part of the model.

Identification of complementary solutions – since organizations tend not to perform changes to an ERP system, there is a need to identify complementary solutions for processes that are not properly handled by the ERP system. In the other models, there is no need for this activity, since the definition of complementary systems is done as part of the determination of the IS policy of the organization.

Construction of a prototype – in the ERP model, the use of a prototype is a must, due to the complexity and large scope of the system. In the SDLC and application software package models, a prototype is optional, and its objective is to investigate definitional issues when necessary.

Definition of work procedures – this activity is very similar in all models. The difference is that in an ERP model, work procedures are also defined for processes that were changed by the project, even though they are not part of the system.

Users' training – in the SDLC and application software package models, training is performed by the technical staff and the system designers. In an ERP model, training is done by the implementation teams.

The Operation Phase

This phase is similar in all life cycle models, except for the support centers issues. The support provided in the SDLC, application software package and prototyping life cycles is based on a help desk and is technical in nature. An ERP life cycle recognizes the importance of support centers which possess knowledge about the system and work processes and which employ technical personnel and key issues.

SUMMARY AND CONCLUSIONS

This paper has developed a generic model for an ERP system development methodology that combines three structured approaches – the traditional system development life cycle (SDLC), the prototyping approach and the application software package life cycle. Due to the unique characteristics of the ERP system, no one of the three models could be solely used for implementing an ERP system. For this reason, the ERP life cycle model is one that combines main features from these models.

The unique features of the suggested ERP life cycle model are:

- The use of a prototype is an integral part of the implementation.
- The implementation phase is iterative.
- Business process redesign is an integral part of the model.
- Users' involvement is mandatory in all phases of the life cycle.
- The model combines activities that attend to changes in the technological infrastructure of the organization.
- The phase of system and vendor selection is the first phase of the ERP life cycle, preceding the definition phase.
- The involvement of the IS unit in the ERP life cycle is low relative to the other life cycle models. It can be expected that within a few years, after gaining more ERP experience, most IS units will be more involved.
- An ERP life cycle uses the ERP system vendor and external consultants in the definition and implementation phases. In the other models, most of the work is performed by the internal IS unit.

The main contribution of this paper is the development of a generic ERP life cycle model that is independent of industry, market, or a specific ERP system. By using this model, an organization can reduce ERP project risks and increase the probability of project success. The Gartner Group claims that an organization that will adopt a life cycle approach to an ERP system could exercise savings of 10% of total system costs over a period of five years (10).

REFERENCES

1. Ahituv, N., S. Neumann, and H.M. Riley. **Principles of Information Systems for Management, 4th ed.** Dubuque, IA: Business and Educational Technologies, 1994.
2. Alavi, M. "The Evolution of Information System Development Approach: Some Field Observations," **Database,** 15:3, Spring 1984, pp. 19-24.
3. Alter, S. **Information Systems: A Management Perspective, 3rd ed.** Menlo Park, CA: Benjamin/Cummings, 1998.
4. Anderson, D. "Super Users to the Rescue," **CIO Magazine,** November 1997, pp. 33-36.
5. Appleton, E.L. "How to Survive ERP," **Datamation,** 43:3, March 1997, pp. 50-54.
6. Bartholomew, D. "Taking Charge – CEOs Should Take an Active Role in Enterprise-wide IT Projects," **Industry Week,** August 18, 1997, pp. 122-129.
7. Bradley, J. and W.D. Hecht. "Choose the Right ERP Software," **Datamation,** 43:3, March 1997, pp. 56-58.
8. Cale, E.G. and S.E. Eriksen. "Factors Affecting the Implementation Outcome of a Mainframe Software Package: A Longitudinal Study," **Information and Management,** 26:3, 1994, pp. 165-175.
9. Cerveny, R.P., E.J. Garrity, and G.L. Sanders. "The Application of Prototyping to Systems Development: A Rationale and Model," **Journal of Management Information Systems,** 3:2, Fall 1986, pp. 52-62.
10. Daily, A. "ERP and Supply Chain Management from Today to 2002," **Proceedings of the Gartner Conference on Information Technology for the 21st Century,** Tel-Aviv, Israel, May 1998, pp. 128-145.
11. Davenport, T.H. and J.E. Shoft. "The New Industrial Engineering: Information Technology and Business Process Redesign," **Sloan Management Review,** 31:3, Summer 1990, pp. 11-27.
12. Franz, C.R. "User Leadership in the Systems Development Life Cycle: A Contingency Model," **Journal of Management Information Systems,** 2:2, Fall 1985, pp. 5-25.

13. Gross, P.H.B. and M.J. Ginzberg. "Barriers to the Adoption of Application Software Packages," **Systems, Objectives, Solutions,** 4:3, 198, pp. 211-226.
14. Grover, V., S.R. Jeong, W.J. Kettinger, and J.T.C. Teng. "The Implementation of Business Process Reengineering," **Journal of Management Information Systems,** 12:1, 1995, pp. 31-56.
15. Janson, M.A. "Prototyping for Systems Development: A Critical Appraisal," **MIS Quarterly,** 10:4, December 1985, pp. 305-315.
16. Keen, P.G.W., G.S. Bronsema, and S. Zuboff. "Implementing Common Systems: One Organization's Experience," **Systems, Objectives, Solutions,** 2:3, August 1982, pp. 125-142.
17. Lieber, R.B. "Here Comes SAP," **Fortune,** March 1998, pp. 122-124.
18. Lucas, H.C., E.J. Walton, and M.J. Ginzberg. "Implementing Packaged Software," **MIS Quarterly,** 13:4, December 1988, pp. 537-549.
19. McFarlan, F.W. "Portfolio Approach to Information Systems," **Harvard Business Review,** 59:5, September-October 1981, pp. 142-150.
20. Naumann, J.D. and A.M. Jenkins. "Prototyping: The New Paradigm for Systems Development," **MIS Quarterly,** 6:3, September 1982, pp. 29-44.
21. Neumann, S. **Strategic Information Systems.** New York: Macmillan, 1994.
22. Ragowsky. A., N. Ahituv, and S. Neumann. "Documenting the Benefits an Organization May Gain by Using Information Systems," **Communications of the ACM,** Forthcoming.
23. Ragowsky, A., N. Ahituv, and S. Neumann. "A Model for Identifying the Value and the Importance of an Information System Application," **Information and Management,** 31:5, 1996, pp. 89-102.
24. Schonberger, R.J. "MIS Design: A Contingency Approach," **MIS Quarterly,** 4:1, March 1980, pp. 13-20.
25. Shin, H. and J. Lee. "A Process Model of Application Software Package Acquisition and Implementation," **Journal of Systems and Software,** 32:2, 1996, pp. 57-64.
26. Stein, T. "Keyword: Integration," **Information Week,** September 1997, pp. 73-76.
27. Stevens, T. "Kodak Focuses on ERP," **Industry Week,** August 1997, pp. 130-134.
28. Turban, E., E. McLean, and J.C. Wetherbe. **Information Technology for Management, 2nd ed.** New York: John Wiley and Sons, 1997.
29. Vasilash, G.W. "How To – And How Not To – Implement ERP," **Automotive Manufacturing and Production,** August 1997, pp. 64-65.
30. Zwass, V. **Foundations of Information Systems.** Boston: Irwin/McGraw Hill, 1997.