The Systems Development Life Cycle Basics

Introduction

Organizations must learn how to build and implement systems to remain competitive. Software that is built correctly can support agile organizations and can transform as the organization and its business transforms. Software that effectively meets employee needs will help an organization become more productive and enhance decision making. Software that does not meet employee needs may have a damaging effect on productivity and can even cause a business to fail. Employee involvement along with using the right implementation methodology when developing software is critical to the success of an organization.

LEARNING OUTCOMES

C.1 Summarize the activities associated with the planning phase in the SDLC.

C.2 Summarize the activities associated with the analysis phase in the SDLC.

C.3 Summarize the activities associated with the design phase in the SDLC.

C.4 Summarize the activities associated with the development phase in the SDLC.

C.5 Summarize the activities associated with the testing phase in the SDLC.

C.6 Summarize the activities associated with the implementation phase in the SDLC.

C.7 Summarize the activities associated with the maintenance phase in the SDLC.

Systems Development Life Cycle

The ***systems development life cycle (SDLC)*** is the overall process for developing information systems from planning and analysis through implementation and maintenance. The SDLC is the foundation for all systems development methodologies and there are literally hundreds of different activities associated with each phase in the SDLC. Typical activities include determining budgets, gathering system requirements, and writing detailed user documentation. The activities performed during each systems development project will vary. The SDLC begins with a business need, followed by an assessment of the functions a system must have to satisfy the need, and ends when the benefits of the system no longer outweigh its maintenance costs. This is why it is referred to as a 'lifecycle'. The SDLC is comprised of seven distinct phases: planning, analysis, design, development, testing, implementation, and maintenance. This section takes a detailed look at a few of the more common activities performed during the phases of the systems development life cycle along with common issues facing software development projects (see Figure C.1 and Figure C.2).

Phase 1: Planning

The ***planning phase*** involves establishing a high-level plan of the intended project and determining project goals. Remember that once a project is identified, the first steps involve assigning a project manager and developing a project charter, then guiding the project and initiating the project planning phase (see Chapter 10).

figure C.1
The Systems Development Life Cycle

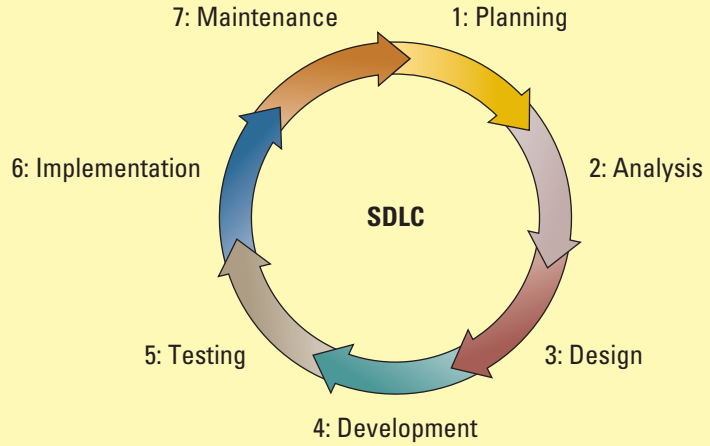


figure C.2
Common Activities Performed During Systems Development

SDLC Phase	Activities
1. Planning	■ Identify and select the system for development ■ Assess project feasibility ■ Develop the project plan
2. Analysis	■ Gather business requirements ■ Create process diagrams ■ Perform a buy versus build analysis
3. Design	■ Design the IT infrastructure ■ Design system models
4. Development	■ Develop the IT infrastructure ■ Develop the database and programs
5. Testing	■ Write the test conditions ■ Perform the system testing
6. Implementation	■ Determine implementation method ■ Provide training for the system users ■ Write detailed user documentation
7. Maintenance	■ Build a help desk to support the system users ■ Perform system maintenance ■ Provide an environment to support system changes

Planning is the first and most critical phase of any systems development effort an organization undertakes, regardless of whether the effort is to develop a system that allows customers to order products over the Internet, determine the best logistical structure for warehouses around the world, or develop a strategic information alliance with another organization. Organizations must carefully plan the activities (and determine why they are necessary) to be successful. The three primary activities involved in the planning phase are:

1. Identify and select the system for development.
2. Assess project feasibility.
3. Develop the project plan.

IDENTIFY AND SELECT THE SYSTEM FOR DEVELOPMENT

Systems are successful only when they solve the right problem or take advantage of the right opportunity. Systems development focuses on either solving a problem or taking advantage of an opportunity. Determining which systems are required to support the strategic goals of an organization is one of the primary activities performed during the planning phase. Typically, employees generate proposals to build new information systems when they are having a difficult time performing their jobs. Unfortunately, most organizations have limited resources and cannot afford to develop all proposed information systems. Therefore, they look to critical success factors to help determine which systems to build.

Determining which systems are required to support the strategic goals of an organization is one of the primary activities performed during the planning phase. Typically, employees generate proposals to build new information systems when they are having a difficult time performing their jobs. Unfortunately, most organizations have limited resources and cannot afford to develop all proposed information systems. Therefore, they look to critical success factors to help determine which systems to build.

figure C.3
Evaluation Criteria for Determining Software Development Projects

Evaluation Criteria	Description
Value chain analysis	The value chain determines the extent to which the new system will add value to the organization. Systems with greater value are given priority over systems with less value.
Strategic alignment	Projects that are in line with the organization's strategic goals and objectives are given priority over projects not in line with the organization's strategic goals and objectives.
Cost/benefit analysis	A cost/benefit analysis determines which projects offer the organization the greatest benefits with the least amount of cost.
Resource availability	Determine the amount and type of resources required to complete the project and determine if the organization has these resources available.
Project size, duration, and difficulty	Determine the number of individuals, amount of time, and technical difficulty of the project.

A ***critical success factor (CSF)*** is a factor that is critical to an organization's success. To determine which system to develop, an organization tracks all the proposed systems and prioritizes them by business impact or critical success factors. This allows the business to prioritize which problems require immediate attention and which problems can wait. Figure C.3 displays possible evaluation criteria for determining which projects to develop.

ASSESS PROJECT FEASIBILITY

A ***feasibility study*** determines if the proposed solution is feasible and achievable from a financial, technical, and organizational standpoint. Typically, an organization will define several alternative solutions that it can pursue to solve a given problem. A feasibility study is used to determine if the proposed solution is achievable, given the organization's resources and constraints in regard to technology, economics, organizational factors, and legal and ethical considerations. Figure C.4 displays the different types of feasibility studies an organization can perform.

DEVELOP THE PROJECT PLAN

Developing a project plan is one of the final activities performed during the planning phase and it is one of the hardest and most important activities. The project plan is the guiding force behind on-time delivery of a complete and successful

figure C.4
Different Types of Feasibility Studies

Types of Feasibility Studies	
Economic feasibility study (often called a **cost-benefit analysis**)	Identifies the financial benefits and costs associated with the systems development project.
Legal and contractual feasibility study	Examines all potential legal and contractual ramifications of the proposed system.
Operational feasibility study	Examines the likelihood that the project will attain its desired objectives.
Schedule feasibility study	Assesses the likelihood that all potential time frames and completion dates will be met.
Technical feasibility study	Determines the organization's ability to build and integrate the proposed system.

system. It logs and tracks every single activity performed during the project. If an activity is missed, or takes longer than expected to complete, the project plan must be updated to reflect these changes. Updating the project plan must be performed in every subsequent phase during the systems development effort.

Phase 2: Analysis

The ***analysis phase*** involves analyzing end-user business requirements and refining project goals into defined functions and operations of the intended system. A good start is essential and the organization must spend as much time, energy, and resources as necessary to perform a detailed, accurate analysis. The three primary activities involved in the analysis phase are:

1. Gather business requirements.
2. Create process diagrams.
3. Perform a buy versus build analysis.

GATHER BUSINESS REQUIREMENTS

Perhaps, the most crucial component of the analysis phase is the development of business requirements. ***Business requirements*** are the detailed set of business requests that the system must meet to be successful. At this point, there is little or no concern with any implementation or reference to technical details. For example, the types of technology used to build the system, such as an Oracle database or the Java programming language, are not yet defined. The only focus is on gathering the true business requirements for the system. A sample business requirement might state, "The system must track all customer sales by product, region, and sales representative." This requirement states what the system must do from the business perspective, giving no details or information on how the system is going to meet this requirement.

The Software Engineering Institute (SEI) defines the objective of requirements management as ensuring customer requirements are the focus of the project from inception to delivery. Any changes must be tracked with corresponding updates made to project documents and plans. ***Requirements management*** is the process of identifying, documenting, communicating, tracking, and managing project requirements as well as changes to those requirements. It is not a single point in time occurrence, but rather must be an ongoing process that stays in lockstep with the development process. Losing sight of requirements is often the first step on the road to over-budget, late projects that do not meet specifications, or end up cancelled.

The most important step in deciding project requirements is obtaining user input on written business requirements. Gathering business requirements is basically conducting an investigation in which users identify all the organization's business needs and take measurements of these needs. Figure C.5 displays different methods organizations use to gather business requirements. A solid understanding of user requirements helps to define application capabilities throughout the project's life cycle. Projects in which users or user groups have a good understanding of their true needs have a better rate of return and lower risk than those that have broader requirements. Often, users find it difficult to articulate their true needs or the IT team misinterprets them. This leads to a compromise during this phase whereby questions remain unanswered and a communication gap develops between the user community and the IT team. It is essential that the project manager, together with the development team, get the business requirements right! The ***requirements definition document*** contains the final set of business requirements, prioritized in order of business importance. The system users review the requirements definition document and determine if they will sign off on the business requirements. ***Sign-off*** is the system users' actual signatures indicating they approve all of the business requirements. One of the first major milestones on the project plan is usually the users' sign-off on business requirements.

figure C.5
Methods for Gathering Business Requirements

Methods for Gathering Business Requirements
Perform a ***joint application development (JAD)*** session where employees meet, sometimes for several days, to define or review the business requirements for the system.
Interview individuals to determine current operations and current issues.
Compile questionnaires to survey employees to discover issues.
Make observations to determine how current operations are performed.
Review business documents to discover reports, policies, and how information is used throughout the organization.

A large data storage company implemented a project called Python whose purpose was to control all the company's information systems. Seven years, tens of millions of dollars, and 35 programmers later, Python was canceled. At the end of the project, Python had over 1,800 business requirements of which 900 came from engineering and were written in order to make the other 900 customer requirements work. By the time the project was canceled, it was unclear what the primary goals, objectives, and needs of the project were. Management should have realized Python's issues when the project's requirements phase dragged on, bulged, and took years to complete. The sheer number of requirements should have raised a red flag.[1]

CREATE PROCESS DIAGRAMS

Once a business analyst takes a detailed look at how an organization performs its work and its processes, the analyst can recommend ways to improve these processes to make them more efficient and effective. ***Process modeling*** involves graphically representing the processes that capture, manipulate, store, and distribute information between a system and its environment. One of the most common diagrams used in process modeling is the data flow diagram. A ***data flow diagram (DFD)*** illustrates the movement of information between external entities and the processes and data stores within the system (see Figure C.6). Process models and data flow diagrams establish the specifications of the system. ***Computer-aided software engineering (CASE)*** tools are software suites that automate systems analysis, design, and development. Process models and data flow diagrams can provide the basis for the automatic generation of the system if they are developed using a CASE tool.

PERFORM A BUY VERSUS BUILD ANALYSIS

An organization faces two primary choices when deciding to develop an information system: (1) it can buy the information system from a vendor or (2) it can build the system itself. ***Commercial off-the-shelf (COTS)*** software is a software package or solution that is purchased to support one or more business functions and information systems. Most customer relationship management, supply chain management, and enterprise resource planning solutions are COTS. Typically, a cost-benefit analysis forms the basis of the buy versus build decision. Figure C.7 displays a few questions an organization must consider when contemplating the buy versus build decision.

Three key factors an organization should also consider when contemplating the buy versus build decision are: (1) time to market, (2) corporate resources, and (3) core competencies (see Figure C.8). Weighing the complex relationship between each of these three variables will help an organization make the right choice.

When making the all-important buy versus build decision consider when the product must be available, how many resources are available, and how the organization's core competencies affect the product. If these questions can be definitely

figure C.6

Sample Data Flow Diagram

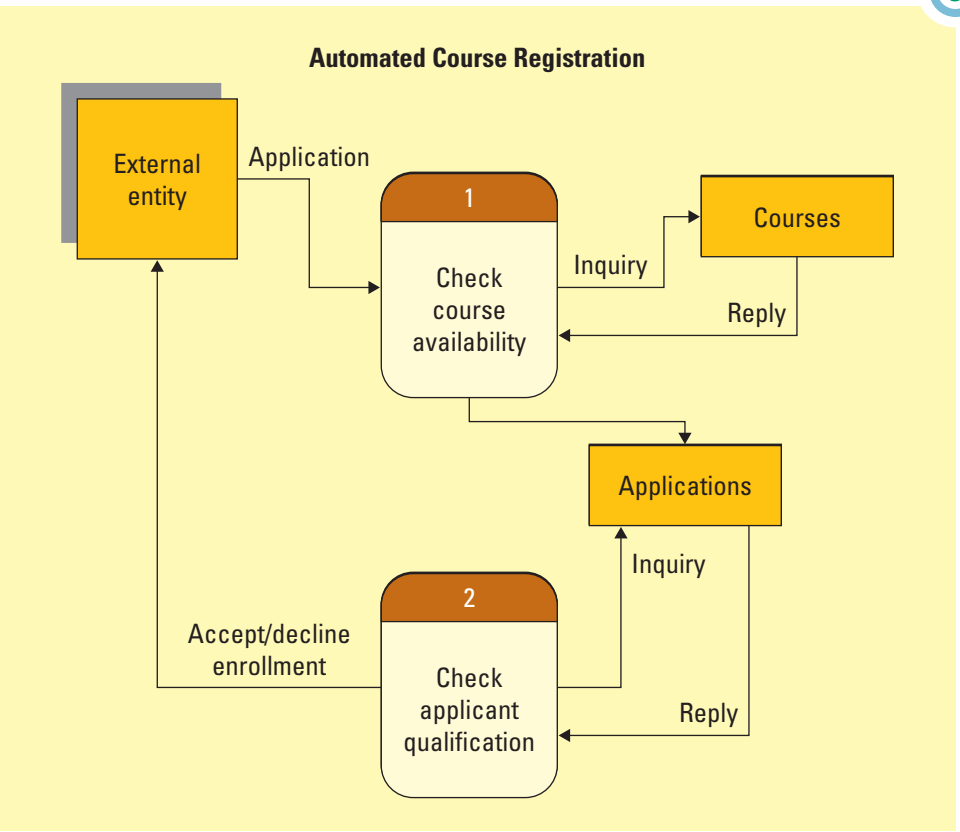


figure C.7

Buy versus Build Decision Questions

Buy versus Build Decision Questions
Do any currently available products fit the organization's needs?
Are unavailable features important enough to warrant the expense of in-house development?
Can the organization customize or modify an existing COTS to fit its needs?
Is there a justification to purchase or develop based on the cost of acquisition?

figure C.8

Key Factors in Buy versus Build Decisions

Three Key Factors in Buy versus Build Decisions	
1. Time to market	If time to market is a priority, then purchasing a good base technology and potentially building on to it will likely yield results faster than starting from scratch.
2. Availability of corporate resources	The buy versus build decision is a bit more complex to make when considering the availability of corporate resources. Typically, the costs to an organization to buy systems such as SCM, CRM, and ERP are extremely high. These costs can be so high—in the multiple millions of dollars—that acquiring these technologies might make the entire concept economically unfeasible. Building these systems, however, can also be extremely expensive, take indefinite amounts of time, and constrain resources.
3. Corporate core competencies	The more an organization wants to build a technical core competency, the less likely it will want to buy.

answered either yes or no, then the answer is easy. However, most organizations cannot answer these questions with a solid yes or no. Most organizations need to make a trade-off between the lower cost of buying a system and the need for a system that meets all of their requirements. Finding a system to buy that meets all an organization's unique business requirements is next to impossible.

Phase 3: Design

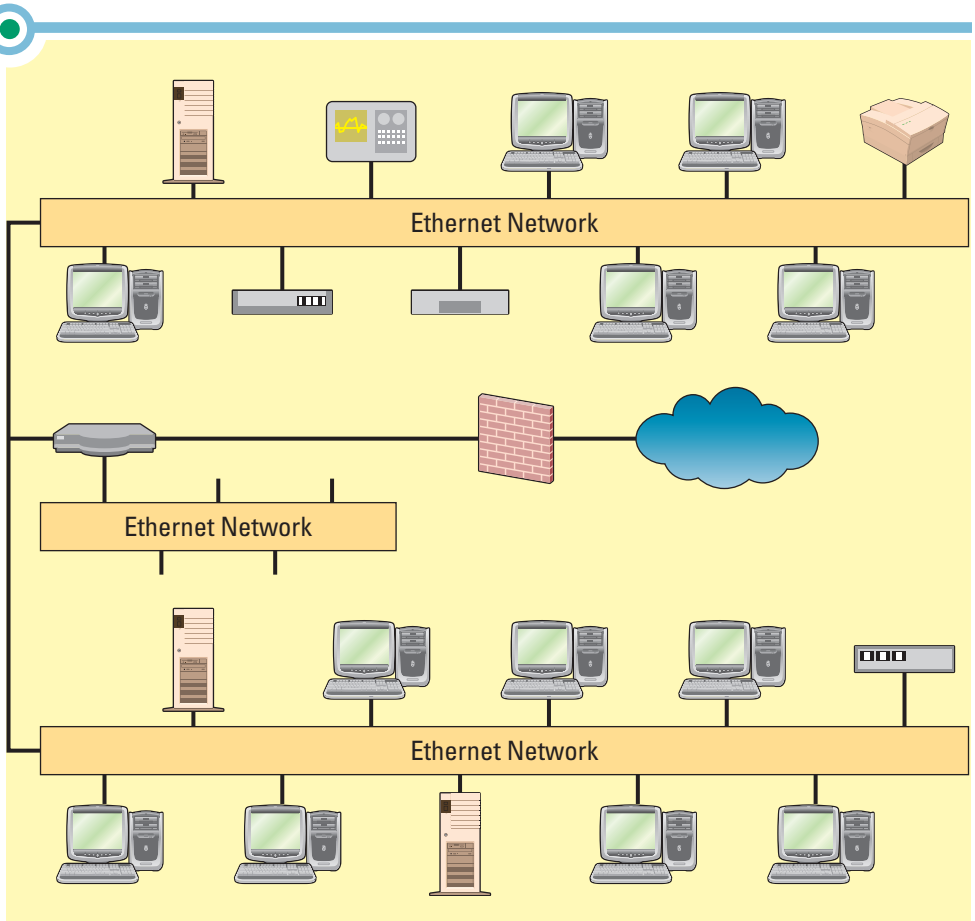
The ***design phase*** involves describing the desired features and operations of the system including screen layouts, business rules, process diagrams, pseudo code, and other documentation. The two primary activities involved in the design phase are:

1. Design the IT infrastructure.
2. Design system models.

DESIGN THE IT INFRASTRUCTURE

The system must be supported by a solid IT infrastructure or chances are the system will crash, malfunction, or not perform as expected. The IT infrastructure must meet the organization's needs in terms of time, cost, technical feasibility, and flexibility. Most systems run on a computer network with each employee having a client and the application running on a server. During this phase, the IT specialists recommend what types of clients and servers to buy including memory and storage requirements, along with software recommendations. An organization typically explores several different IT infrastructures that must meet current as well as future system needs. For example, databases must be large enough to hold the current volume of customers plus all new customers that the organization expects to gain over the next several years (see Figure C.9).

figure C.9
Sample IT Infrastructure



DESIGN SYSTEM MODELS

Modeling is the activity of drawing a graphical representation of a design. An organization should model everything it builds including reports, programs, and databases. Many different types of modeling activities are performed during the design phase including:

- The ***graphical user interface (GUI)*** is the interface to an information system. GUI screen design is the ability to model the information system screens for an entire system using icons, buttons, menus, and submenus.
- ***Data models*** represent a formal way to express data relationships to a database management system (DBMS).
- ***Entity relationship diagram (ERD)*** is a technique for documenting the relationships between entities in a database environment (see Figure C.10).

Phase 4: Development

The ***development phase*** involves taking all of the detailed design documents from the design phase and transforming them into the actual system. The two primary activities involved in the development phase are:

1. Develop the IT infrastructure.
2. Develop the database and programs.

DEVELOP THE IT INFRASTRUCTURE

The platform upon which the system will operate must be built before building the actual system. In the design phase, an organization creates a blueprint of the proposed IT infrastructure displaying the design of the software, hardware, and telecommunication equipment. In the development phase, the organization purchases and implements the required equipment to support the IT infrastructure.

Most new systems require new hardware and software. It may be as simple as adding memory to a client or as complex as setting up a wide area network across several states.

DEVELOP THE DATABASE AND PROGRAMS

Once the IT infrastructure is built, the organization can begin to create the database and write the programs required for the system. IT specialists perform these functions and it may take months or even years to design and create all the needed elements to complete the system.

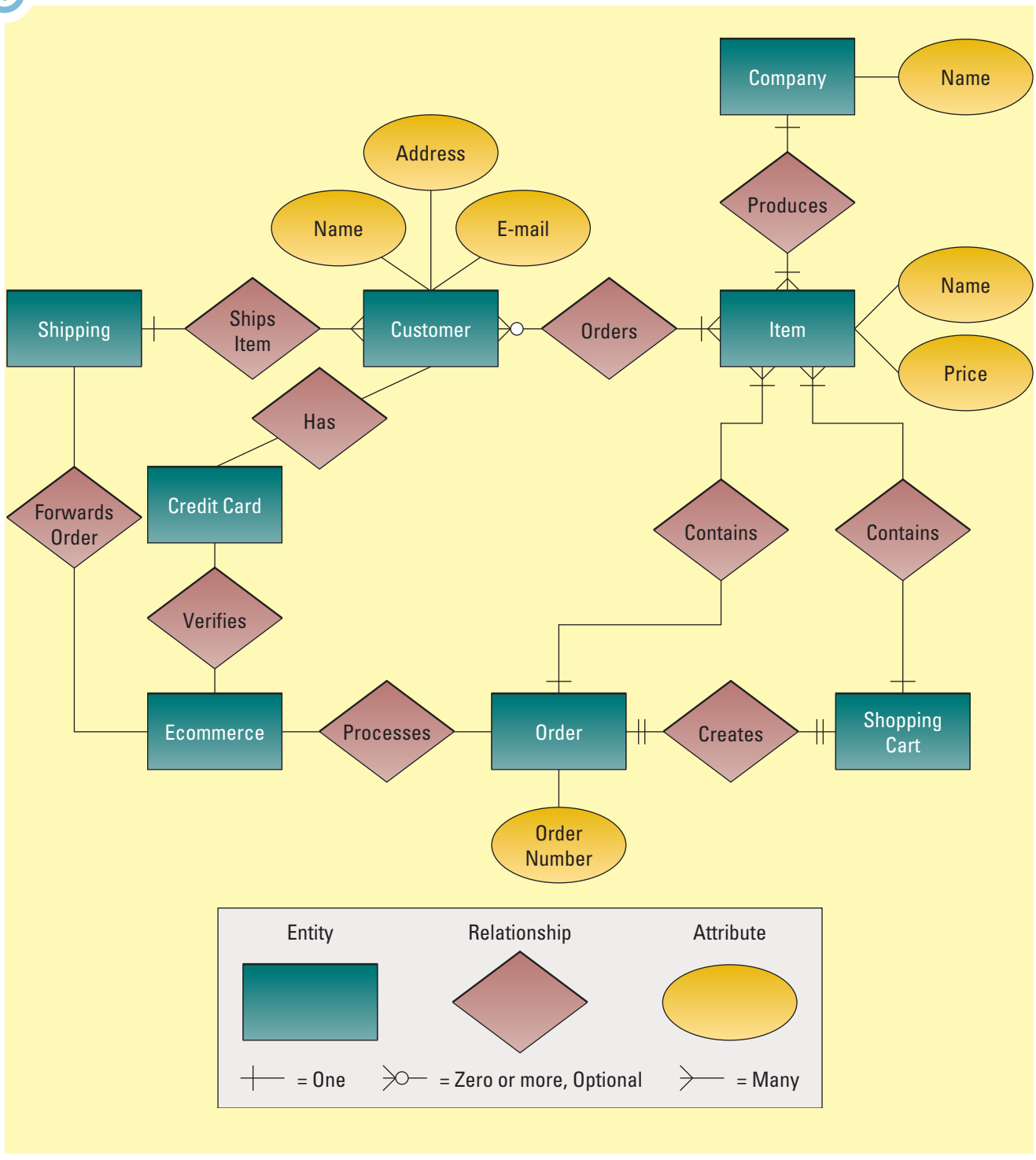
Phase 5: Testing

According to a report issued in June 2003 by the National Institute of Standards and Technology (NIST), defective software costs the U.S. economy an estimated $59.5 billion each year. Of that total, software users incurred 64 percent of the costs and software developers 36 percent. NIST suggests that improvements in testing could reduce this cost by about a third, or $22.5 billion, but that unfortunately testing improvements would not eliminate all software errors.[2]

The ***testing phase*** involves bringing all the project pieces together into a special testing environment to test for errors, bugs, and interoperability, in order to verify that the system meets all the business requirements defined in the analysis phase. The two primary activities involved in the testing phase are:

1. Write the test conditions.
2. Perform the system testing.

figure C.10 Sample Entity Relationship Diagram



WRITE THE TEST CONDITIONS

Testing is critical. An organization must have excellent test conditions to perform an exhaustive test. ***Test conditions*** are the detailed steps the system must perform along with the expected results of each step. Figure C.11 displays several test conditions for testing user log-on functionality in a system. The tester will execute each test condition and compare the expected results with the actual results in order to verify that the system functions correctly. Notice in Figure C.11 how each test

figure C.11
Sample Test Conditions

Test Condition Number	Date Tested	Tester	Test Condition	Expected Result	Actual Result	Pass/ Fail
1	1/1/05	Emily Hickman	Click on System Start Button	Main Menu appears	Same as expected result	Pass
2	1/1/05	Emily Hickman	Click on Logon Button in Main Menu	Logon Screen appears asking for Username and Password	Same as expected result	Pass
3	1/1/05	Emily Hickman	Type Emily Hickman in the User Name Field	Emily Hickman appears in the User Name Field	Same as expected result	Pass
4	1/1/05	Emily Hickman	Type Zahara 123 in the password field	XXXXXXXXX appears in the password field	Same as expected result	Pass
5	1/1/05	Emily Hickman	Click on O.K. button	User logon request is sent to database and user name and password are verified	Same as expected result	Pass
6	1/1/05	Emily Hickman	Click on Start	User name and password are accepted and the system main menu appears	Screen appeared stating logon failed and username and password were incorrect	Fail

condition is extremely detailed and states the expected results that should occur when executing each test condition. Each time the actual result is different from the expected result, a "bug" is generated and the system goes back to development for a bug fix.

Test condition 6 in Figure C.11 displays a different actual result than the expected result because the system failed to allow the user to log on. After this test condition fails, it is obvious that the system is not functioning correctly and it must be sent back to development for a bug fix.

A typical system development effort has hundreds or thousands of test conditions. Every single test condition must be executed to verify that the system performs as expected. Writing all the test conditions and performing the actual testing of the software takes a tremendous amount of time and energy. Testing is critical to the successful development of any system.

PERFORM THE SYSTEM TESTING

System developers must perform many different types of testing to ensure that the system works as expected. System developers must perform many different types of testing to ensure that the system works as expected. Often, the development team asks testers to 'pound' the system or 'hit the system hard'. Stressing the system and attempting to actually cause a failure, often via unorthodox methods, is critical in determining the viability of the system and unforeseen problems. Figure C.12 displays the more common types of tests performed during the testing phase. Standish Group's research shows that coding errors and programming bugs cause the vast majority of all application outages well above hardware, network, and database failures. On the other hand, quality assurance is frequently the first area that is cut back when deadlines are missed. This, of course, increases the problem.

figure C.12
Types of Tests Performed During the Testing Phase

Types of Tests Performed During the Testing Phase	
Application (or system) testing	Verifies that all units of code work together and the total system satisfies all of its functional and operational requirements.
Backup and recovery testing	Tests the ability of an application to be restarted after failure.
Documentation testing	Verifies that the instruction guides are helpful and accurate.
Integration testing	Exposes faults in the integration of software components or software units.
Regression testing	Determines if a functional improvement or repair to the system has affected the other functional aspects of the software.
Unit testing	Tests each unit of code as soon as the unit is complete to expose faults in the unit regardless of its interaction with other units.
User acceptance testing (UAT)	Determines whether a system satisfies its acceptance criteria, enabling the customer to decide whether or not to accept a system.

An error caught within the development process is 10 to 100 times cheaper to correct than a bug found during the application's operation.

Phase 6: Implementation

The ***implementation phase*** involves placing the system into production so users can begin to perform actual business operations with the system. The implementation phase is also referred to as 'delivery'. The implementation phase is comprised of two activities: training and conversion. Each of these activities include multiple part tasks such as writing detailed user documentation, determining the conversion method, and providing training for system users. How and what time during the phase these tasks occurs, often depends upon the conversion method selected. For example, for a plunge conversion, all training must take place prior to the conversion. Alternatively, during a parallel conversion, training can be offered at scheduled intervals as the new system is rolled out. Also, the complexity and comprehensive nature of the new system can dictate timing and steps necessary to deliver or implement the system. The two primary activities of the implementation phase include:

1. System Training
2. Implementation Method

SYSTEM TRAINING

System users require ***user documentation*** that details how to use the system. User documentation typically is provided with the new system. The user documentation must be presented in language that is easy to understand and digest with examples and instruction that are simple to follow. Including graphics and sample problems is recommended. It also must be easily accessed and made available (and updated) for as long as the system is live. Delivery methods can include an internal online system such as a private intranet or via hard copy in the form of manuals. Developing and distributing detailed user documentation is a cost related to the overall project and must not be overlooked or excluded from cost estimates. System users find it extremely frustrating to have a new system without documentation.

METHODS FOR TRAINING SYSTEM USERS

By simply training users, organizations can speed implementation, realize cost and productivity benefits early, create positive 'buzz', and alleviate potential, nagging conversion problems. Most help desk professionals will agree that nothing is more wasteful than to answer the same question many times a day that could have easily been addressed through routine user training. An organization must provide training for the system users. The two most popular types of training are self-paced and group training. The most common form of self-paced training is online training. ***Online training*** is delivered via the web, a CD-ROM, DVD, or other static media a user can access instantly and as needed. Online training is cost effective because it can be deployed fairly easily and to the entire user community simultaneously and because it is self-directed and does not unnecessarily tap additional needed personnel. System users schedule their own training, on their own computers, at their own pace. They are responsible for learning the system by clarifying questions as they go. Mastery of a new system via self-paced training can be made evident by including tests or quizzes as a condition of training completion.

Group training is just that—learning the new system by training groups of users rather than individuals. Common forms of group training include train-the-trainer, workshop training, and online group training using technology such as a product called, WebEx. ***Train-the-trainer*** involves developing a few system experts and deploying them throughout the organization to train others. This method often frees up the development team to proceed to the next project without being 'on call' to troubleshoot user inquiries. Trainers often can serve as a conduit to the development team providing feedback regarding system training and conversion. ***Workshop training*** is set in a classroom environment and led by an instructor. Workshop training is recommended for difficult systems where the system users require one-on-one time with an individual instructor. ***Online group training,*** such as a WebEx presentation, is extremely effective in training system users who are geographically dispersed. The training is conducted by a knowledgeable presenter using both phones and computers. Users call into the session, are connected to the presentation, are guided through the training by the presenter, and follow the training on their own computer. Users can either ask questions real-time or send in questions to be addressed at a later time.

IMPLEMENTATION METHOD

An organization must choose the conversion method that most clearly meets its business objectives and can realistically be deployed to ensure a successful system implementation. There are four primary implementation methods an organization can use (see Figure C.13). For example, if a company is in an industry where governmental compliance is mandatory and a system will be out of compliance by a certain date and that date has arrived, a company can only choose a plunge conversion. Or, let's say a company has few resources available for system monitoring and user training. It probably is wise for it to choose a phased conversion so it can effectively monitor the system and train users.

Phase 7: Maintenance

The ***maintenance phase*** involves performing changes, corrections, additions, and upgrades to ensure the system continues to meet the business goals. This phase continues for the life of the system because the system must change as the business evolves and its needs change, demanding constant monitoring, supporting the new system with frequent minor changes (for example, new reports or information capturing), and reviewing the system to be sure it is moving the organization toward its strategic goals. Once a system is in place, it must change as the

figure C.13 Primary Conversion Methods

Method	Description	Pros	Cons	Troubleshooting
1. Parallel Conversion	Using both the old and new systems until it is evident that the new system performs correctly.	Eliminates the risk of operational 'downtime' in the event of a new system failure. Gives users time to acclimate to the new system.	Expensive to run two systems and has high human capital risk as it allows negative reactions to the new system to percolate through the organization. The benefits of the system are not immediately realized.	Establish a 'drop dead' date for moving all users to the new system—do not delay the complete conversion longer than is necessary. Report benefits as soon as they are realized. Use 'corporate champions' to rally the troops as they convert.
2. Phased Conversion	Implementing the new system in phases (e.g., accounts receivables then accounts payable) until it is evident that the new system performs correctly and then implementing the remaining phases of the new system.	Reduces the risks associated with a whole system failure. Gives users time to master a single phase or module at one time which improves user satisfaction with the system.	The project team and user community often experience system fatigue—the feeling of just wanting it to be over. This effects morale and the overall sense of excitement that should be equated with a new system that is expected to deliver positive results. Again, the benefits of the system are not immedi-ately realized.	Ensure the phases are released quickly with all associated training, collateral, and components. It is OK to release the next phase even if you have a few latent users. They will adapt quickly once they realize they must! Be specific regarding the schedule and the finish date for the system implementation.
3. Pilot Conversion	Having only a small group of people use the new system until it is evident that the new system performs correctly and then adding the remaining people to the new system.	Allows time to refine the process and troubleshoot potential pitfalls. Problems are obvious to only a small user group which helps to maintain the enthusiasm and momentum of the implementation corporate-wide.	Pilot groups can be perceived as being 'more important' than others creating negative feelings about the implementation causing training to be a tough process. Getting users on board is emotionally difficult at times. Once again, benefits of the system implementation are delayed.	Clearly communicate the benefits of pilot testing to all users. Establish the conversion schedule addressing timeframe for when each user groups can expect to be up and running on the new system.
4. Plunge Conversion	Discarding the old system completely and immediately using the new system.	Reduces costs by only deploying resources to the new system. Eliminates confusion among users as to which system to use and when. Hastens training as users need to get up to speed quickly. Users perceive leadership as intentional and system benefits can be realized immediately.	Has the highest amount of technology risk—the system can fail and stall operations. Users become aggravated and dismayed by the failure. Sentiments such as, 'at least we knew the old system worked', might arise lowering morale.	Calculate the potential risk and develop contingency plans to mitigate it. Ensure leadership is on board—they understand the risks and can communicate the contingency plans to staff to generate buy-in and decrease fears about the conversion—in essence, eliminate panic if the system fails. Reward development team with comp time, extra pay, or days off if they spend extra time ensuring a successful conversion.

organization changes. The three primary activities involved in the maintenance phase are:

1. Build a help desk to support the system users.
2. Perform system maintenance.
3. Provide an environment to support system changes.

BUILD A HELP DESK TO SUPPORT THE SYSTEM USERS

The most common method for supporting systems users is to establish a help desk. A ***help desk*** is a group of people who respond to internal system user questions. Typically, internal system users have a help desk extension or phone number they call when they have issues or questions about the system. Staffing a help desk that answers internal user questions is an excellent way to provide comprehensive support for new systems. It also can serve as a feedback mechanism to the development team to relay common problems that need to be addressed; to report obvious oversights in training; and to identify and report widespread systems errors called 'bugs'. New hire and regularly recurring refresher training also serve to support system users.

PERFORM SYSTEM MAINTENANCE

Maintenance is fixing or enhancing an information system. Many different types of maintenance must be performed on the system to ensure it continues to operate as expected. These include:

- **Adaptive maintenance—**making changes to increase system functionality to meet new business requirements.
- **Corrective maintenance—**making changes to repair system defects.
- **Perfective maintenance—**making changes to enhance the system and improve such things as processing performance and usability.
- **Preventive maintenance—**making changes to reduce the chance of future system failures.

PROVIDE AN ENVIRONMENT TO SUPPORT SYSTEM CHANGES

As changes arise in the business environment, an organization must react to those changes by assessing the impact on the system. It might well be that the system needs to adjust to meet the ever-changing needs of the business environment. If so, an organization must modify its systems to support the business environment.

A ***change management system*** includes a collection of procedures to document a change request and define the steps necessary to consider the change based on the expected impact of the change. Most change management systems require that a change request form be initiated by one or more project stakeholders (users, customers, analysts, developers). Ideally, these change requests are reviewed by a ***change control board (CCB)*** responsible for approving or rejecting all change requests. The CCB's composition typically includes a representative for each business area that has a stake in the project. The CCB's decision to accept or reject each change is based on an impact analysis of the change. For example, if one department wants to implement a change to the software that will increase both deployment time and cost, then the other business owners need to agree that the change is valid and that it warrants the extended time frame and increased budget.

Software Problems Are Business Problems

Only 28 percent of projects are developed within budget and delivered on time and as promised, says a Standish Group report. The primary reasons for project failure are:

- Unclear or missing business requirements.
- Skipping SDLC phases.
- Failure to manage project scope.
- Failure to manage project plan.
- Changing technology.[3]

UNCLEAR OR MISSING BUSINESS REQUIREMENTS

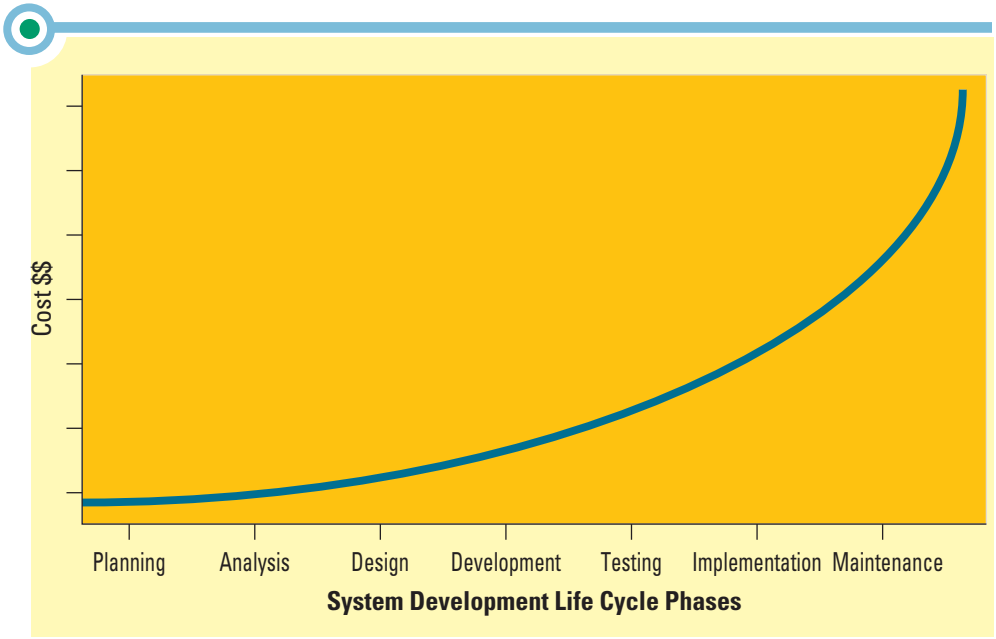
The most common reason systems fail is because the business requirements are either missing or incorrectly gathered during the analysis phase. The business requirements drive the entire system. If they are not accurate or complete, the system will not be successful.

It is important to discuss the relationship between the SDLC and the cost for the organization to fix errors. An error found during the analysis and design phase is relatively inexpensive to fix. All that is typically required is a change to a Word document. However, exactly the same error found during the testing or implementation phase is going to cost the organization an enormous amount to fix because it has to change the actual system. Figure C.14 displays how the cost to fix an error grows exponentially the later the error is found in the SDLC.

SKIPPING SDLC PHASES

The first thing individuals tend to do when a project falls behind schedule is to start skipping phases in the SDLC. For example, if a project is three weeks behind in the development phase, the project manager might decide to cut testing down from six weeks to three weeks. Obviously, it is impossible to perform all the testing in half the time. Failing to test the system will lead to unfound errors, and chances are high that the system will fail. It is critical that an organization perform all phases in the SDLC during every project. Skipping any of the phases is sure to lead to system failure.

figure C.14
The Cost of Finding Errors



FAILURE TO MANAGE PROJECT SCOPE

It is natural for users to want everything—they would be especially happy if their computer even got up and fetched them a cup of coffee! As such, it is imperative to reduce or minimize features and functions. ***Feature creep*** occurs when developers add extra features not part of the initial requirements. ***Scope*** is a collection of requirements. ***Scope creep*** occurs when the scope of the project increases beyond the initial scope statement. Scope creep and feature creep are difficult to manage and can easily cause a project to fall behind schedule, increase costs, or both. Often, scope creep occurs, at times, innocently enough by a simple hallway discussion of adding 'bells and whistles'. The key to reducing time and money is constraining scope to just those elements that are absolutely necessary. Optimizing scope minimizes risk and maximizes return. Project managers can discourage an unwieldy scope by keeping the executive sponsor informed and using a formal change management or change control system.

FAILURE TO MANAGE PROJECT PLAN

Managing the project plan is one of the biggest challenges during systems development. The project plan is the road map the organization follows during the development of the system. Developing the initial project plan is the easiest part of the project manager's job. Managing and revising the project plan is the hard part. The project plan is a living document since it changes almost daily on any project. Failing to monitor, revise, and update the project plan can lead to project failure.

CHANGING TECHNOLOGY

Many real-world projects have hundreds of business requirements, take years to complete, and cost millions of dollars. Gordon Moore, co-founder of Intel Corporation, observed in 1965 that chip density doubles every 18 months. This observation, known as Moore's law, simply means that memory sizes, processor power, and so on, all follow the same pattern and roughly double in capacity every 18 months. As Moore's law states, technology changes at an incredibly fast pace; therefore, it is possible to have to revise an entire project plan in the middle of a project as a result of a change in technology. Technology changes so fast that it is almost impossible to deliver an information system without feeling the pain of changing technology. Change management is all about setting realistic expectations. The importance of this critical factor cannot be understated. Almost nothing can cause a misalignment between expectations and deliverables more quickly than a failure to manage change.

Key Terms

Analysis phase C.5
Business requirement C.5
Change control board (CCB) C.15
Change management system C.15
Commercial off-the shelf (COTS) C.6
Computer-aided software engineering (CASE) C.6
Critical success factor (CSF) C.4
Data flow diagram (DFD) C.6
Data model C.9
Design phase C.8
Development phase C.9
Entity relationship diagram (ERD) C.9
Feasibility study C.4
Feature creep C.17
Graphical user interface (GUI) C.9
Group Training C.13
Help desk C.15
Implementation phase C.12
Joint application development (JAD) C.6
Maintenance C.15
Maintenance phase C.13
Modeling C.9
Online group training C.13
Online training C.13
Planning phase C.2
Process modeling C.6
Requirements definition document C.5
Requirements management C.5
Scope C.17
Scope creep C.17
Sign-off C.5
Systems development life cycle (SDLC) C.2
Test condition C.10
Testing phase C.9
Train-the-Trianer C.13
User documentation C.12
Workshop training C.13

Making Business Decisions

1. Missing phases in the systems development life cycle

Hello Inc. is a large concierge service for executives operating in Vancouver, Calgary, and Toronto. The company performs all kinds of services from dog walking to airport transportation. Your manager, Dan Martello, wants to skip the testing phase during the company's financial ERP implementation. Dan feels that since the system came from a vendor it should work correctly. To meet the project's looming deadline he wants to skip the testing phase. Draft a memo explaining to Dan the importance of following the SDLC and the ramifications to the business if the financial system is not tested.

2. Refusing to sign off

You are the primary client on a large extranet development project. After carefully reviewing the requirements definition document, you are positive that there are missing, ambiguous, inaccurate, and unclear requirements. The project manager is pressuring you for your sign-off since he has already received sign-off from five of your co-workers. If you fail to sign off on the requirements, you are going to put the entire project at risk since the time frame is nonnegotiable. What would you do? Why?

3. Saving failing systems

Crik Candle Company manufactures low-end candles for restaurants. The company generates over $40 million in annual revenues and has more than 300 employees. You are in the middle of a large multimillion-dollar supply chain management implementation. Your project manager has just come to you with the information that the project might fail for the following reasons:

- Several business requirements were incorrect and the scope has to be doubled.
- Three developers recently quit.
- The deadline has been moved up a month.

Develop a list of options that your company can follow to ensure the project remains on schedule and within budget.

4. Feasibility studies

John Lancert is the new managing operations director for a large construction company, LMC. John is currently looking for an associate who can help him prioritize the 60 proposed company projects. You are interested in working with John and have decided to apply for the job. John has asked you to compile a report detailing why project prioritization is critical for LMC, along with the different types of feasibility studies you would recommend that LMC use when determining which projects to pursue.

Notes

1. "Python Project Failure," www.systemsdev.com, accessed November 14, 2003.
2. Gary McGraw, "Making Essential Software Work," *Software Quality Management,* April 2003, www.sqmmagazine.com, accessed November 14, 2003.
3. www.standishgroup.com, accessed November 14, 2003.