Chapter 1
The Systems Development Environment

Chapter Overview

Chapter 1 is an overview of the systems development process, as well as an overview of the textbook. This chapter introduces students to the modern approach to systems analysis and design using various methodologies. Students are introduced to several systems development components, including the process and data-oriented approaches to systems development; different types of information systems; and the systems development life cycle.

This textbook is intended primarily for juniors taking a core course in the information systems major, although the book can be adapted for a similar course at the junior college level or for a two-course sequence on analysis and design. Often students are not familiar with the systems development process, different organizational components, or how these components work together. This chapter provides the general organizational context in which systems development takes place.

The text uses the Systems Development Life Cycle (SDLC) methodology (including its associated problems with the traditional waterfall approach) to introduce students to the structured approach in creating new systems. The student is also introduced to other methodologies such as Computer-Aided Systems Engineering (CASE), the Agile Methodologies, eXtreme Programming, Object Oriented Analysis and Design (OOAD), and the Rational Unified Process (RUP). The text compares and contrast the new with more traditional methods in an effort to show both the advantages and limitations of these methods.

Chapter 1 introduces students to CASE usage in organizations, CASE components, and visual and emerging development tools. CASE tools are used to apply an engineering approach to systems development and can support each phase of the SDLC. The purpose of CASE is to make it easier for an organization to enact a single design philosophy across many projects, systems, and stakeholders. CASE provides support for all SDLC activities.

Instructional Objectives

Specific student learning objectives are included at the beginning of the chapter. *From an instructor's point of view, the objectives of this chapter are to:*

1. Define and discuss the modern approach to systems analysis and design from an organizational perspective incorporating techniques, tools, and methodologies.

2. Explain how an organization's objectives, structure, and processes are essential in the development of systems to meet their needs.

3. Explain that the SDLC process is not sequential but cyclical and that the order is adaptable as required for different projects; also to emphasize that often analysts and designers may go backwards to the previous step to complete unfinished products or to correct errors or omissions discovered in the next phase.

4. Explain the difference between the logical design and the physical design as it relates to systems development.

5. Discuss the Microsoft Security Development Lifecycle (SDL) and the need to incorporate system security at the beginning of system development.

6. Discuss the problems with the waterfall SDLC and explain the different approaches analysts, designers and developers have developed to improve the Systems Analysis and Design process.

7. Discuss computer-aided software engineering (CASE) tools and their application to the SDLC, specifically how the tools are applied and in which phase of the SDLC.

8. Discuss Agile methodologies and eXtreme programming and how these compare to the traditional Systems Development Life Cycle (SDLC).

9. Explain and discuss object-oriented analysis and design (OOAD) and the Rational Unified Process (RUP).

10. Show students that the life cycle is a flexible basis for systems analysis and design and that it can support many different tools and techniques, such as Agile methodologies and eXtreme Programming.

11. Compare and contrast the various development approaches introduced in Chapter 1 and depict how they all use an iterative approach.

12. Finally explain that the boundaries and divisions of the 5 steps in Figure 1-2 when imposed to explain the steps are neither hard nor fast and that in many real-world situations phases or sub-phases may be combined to improve efficiency and understanding. The cycle is an organizing and guiding principle; however in companies and software development teams will adapt it to suit their needs for specific projects.

Classroom Ideas

1. Figure 1-1 depicts that methodologies, techniques, and tools drive organizational approaches to systems analysis and design. Ask students to identify the names of methodologies, techniques, and tools. List them on the board under the heading that they suggest; then after they have identified 5 or 6 in each heading review and emphasize the differences between the three and move any from an incorrect category to the correct one and explain why it is one and not the other.

2. When introducing the systems development life cycle model featured in the textbook, discuss other life cycle models using actual ones from existing organizations. Show that the basic model presented (Planning, Analysis, Design, Implementation, and Maintenance) are broken down into smaller phases by many companies but that in the end that could be categorized into one of the basic five explained. This reinforces to students that no one standard life cycle model exists and the model they will use as a systems analyst will likely differ from the textbook's life cycle model. The point is that the life cycle represents activities that must be

done; and the phases are a way to introduce, in an organized way, the methods, techniques, tools, and skills necessary for successful systems analysis and design.

3. Provide a brief overview of the activities and outputs from each of the five life cycle phases, based on your own experience or from reading the rest of the textbook. Table 1-1 summarizes the outputs or products of each phase based on the in-text descriptions.

4. This chapter briefly introduces students to CASE tools. It would be best to explain how CASE tools have met with resistance in some organizations and a brief discussion of the advantages and disadvantages of CASE would be appropriate here. Major emphasis should be placed on the increasing use of a CASE repository and that the other tools are used depending upon the complexity of the project and the skill set and training of the organizational developers. Table 1.2 lists examples of CASE usage within the SDLC. Many managers are reluctant to ramp up use of CASE tools due to their difficulty and the high end platforms that many require for support.

5. It would be great if you have any CASE tools available that you could demonstrate to the students by showing them how they could integrate into the SDLC.

6. Figure 1-11 illustrates the RUP life cycle. Discuss RUP, its benefits and drawbacks as it relates to OOAD. Discuss the differences between RUP and the traditional SDLC. Table 1-3 presents the Agile Manifesto.

7. Ask students to compare Agile methodologies to traditional SDLC (see Table 1-4 Five Critical Factors that Distinguish Agile and Traditional Approaches to Systems Development). Introduce a case study project where Agile methodologies were employed. Ask students to identify problems that the project ran into using Agile methodologies as well as any benefits gained by this approach.

8. This chapter introduces eXtreme programming. If your students have sufficient background, assign students to programming pairs and have them work on a small programming problem, including testing. Ask students to report upon their experience.

9. Discuss IBM's Rational Unified Process (RUP) shown in Figure 1-13. This Web site http://www-01.ibm.com/software/awdtools/rup/ / should help with background information.

Answers to Key Terms

Suggested answers are provided below. These answers are presented top-down, left to right

1.6. Information systems analysis and design		1.18. Systems development methodology
1.2. Application software		1.17. Systems development life cycle (SDLC)
1.16. Systems analyst		1.14. Planning
1.1. Analysis		1.12. Object-oriented analysis and design (OOAD) (RAD)
1.4. Design		1.10. Object
1.8. Logical design		1.7. Inheritance

1.13. Physical design		1.11. Object class
1.5. Implementation		1.15. Rational Unified Process (RUP) an example of OOSAD
1.9. Maintenance		
1.3. Computer-aided software engineering tools (CASE)		

Answers to Review Questions

1.19. Information systems analysis and design is the complex organizational process whereby computer-based information systems are developed and maintained.

1.20. In the early years of computing, analysis and design were considered an art. However, with the growing importance and changing nature of information technology and its usage in the work environment, work methods have evolved, making analysis and design a disciplined process. The analysis and design of computer-based information systems began in the 1950s with emphasis placed on automating existing processes. All applications were developed in machine language or assembly language and developed from scratch. The 1960s saw the first procedural, or third-generation languages, become available. Computers were still large and expensive and storage was a premium. In the 1970s, systems development became more disciplined as many people worked to make it more like engineering. In the 1980s, microcomputers became key organizational tools; the software industry expanded greatly; fourth-generation languages were used more and more to write applications, and CASE tools were developed. In the 1990s, the focus shifted to system integration, and developers were using visual programming environments to design user interfaces. Databases began residing on servers, as well as the application logic. Companies began purchasing enterprise-wide systems and more and more systems development focused on the Internet, particularly the Web. The current focus is on Web-based systems development and wireless components. Additionally, many system implementations use a three-tier design. Currently, companies may assemble their systems using off-the-shelf components or by using application service providers.

1.21. The five systems development life cycle phases are planning, analysis, design, implementation, and maintenance. During the planning phase, an organization's total information system needs are identified, analyzed, prioritized, and arranged. During the analysis phase, requirements are gathered from users. The requirements are then studied and organized with any redundancies eliminated. The output of this phase is a solution recommended by the analysis team. During the design phase, the description of the recommended solution is converted into logical and then physical system specifications. During the implementation phase, the information system is coded, tested, installed, and supported in the organization. During the maintenance phase, the system is systematically repaired and improved. Another problem was that roles of system users or customers was narrowly defined with users relegated to the requirements determination or analysis phase where it was assumed that all requirements could be specified in advance. In addition, hard dates were set for the early phases and were judged successful if the dates were met leaving little time to incorporate important changes. The end result of these problems is that the focus on deadlines led to systems that did not match users'

1.22. There have been several problems with the traditional waterfall SDLC identified in the literature. One is that the "downhill" nature of the SDLC process treats each phase as separate and complete unto itself and feedback is often ignored resulting in locking users into requirements that had been

previously determined, even though those requirements might have changed. Another problem was that roles of system users or customers was narrowly defined with users relegated to the requirements determination or analysis phase where it was assumed that all requirements could be specified in advance. In addition, hard dates were set for the early phases and were judged successful if the dates were met leaving little time to incorporate important changes. The end result of these problems is that the focus on deadlines led to systems that did not match users' needs and that required increasing development costs.

1.23. CASE tools are software to provide automated support for some portion of the systems development process.

1.24. The general types and components of CASE tools are diagramming tools, computer display and report generators, analysis tools, a central repository, documentation generators, and code generators. Diagramming tools enable system process, data, and control structures to be represented graphically. Computer display and report generators help prototype how systems "look and feel" to users. Analysis tools automatically check for incomplete, inconsistent, or incorrect specifications in diagrams, forms, and reports. A central repository enables the integrated storage of specification, diagrams, reports, and project management information. Documentation generators produce technical and user documentation in standard formats. Code generators enable the automatic generation of program and database definition code directly from the design documents, diagrams, forms, and reports. Depending on the CASE product, CASE may provide security features, version control, import and export facilities, and facilities for backup and recovery, user account management, and usage accounting. The repository is likely the most important component of a comprehensive CASE system because it provides the mechanism through which tool and activity integration can occur and can be utilized by all team project members, promoting a single view of the project.

1.25. CASE tools are used to support a wide variety of SDLC activities. CASE tools can be used to help in multiple phases of the SDLC such as project identification and selection, project initiation and planning, analysis, design, and implementation and maintenance. This is all because of the database repository used for product and tool integration.

1.26. Agile methodologies promote a self-adaptive software development process. While other methodologies focus on roles that individuals play in a project team, Agile methodologies focus more on the individual. As software is developed, the process used to develop it is refined and improved through a review process done by the development team through iteration.

1.27. eXtreme programming is an approach to software development distinguished by short development cycles, an incremental planning approach, a focus on automated tests written by programmers and customers to monitor the development process, and reliance on an evolutionary approach to development that lasts throughout the lifetime of the system. This methodology uses an evolutionary approach to software development. Coding and testing are part of the same process and are done by a two person programming team. Code is tested shortly after it is written and integrated into the system within a few hours of being written. All phases of the life cycle converge into a series of activities based on coding, testing, listening, and designing.

1.28. Agile methods would be more likely to be employed instead of a more engineering-based approach when the project or team is relatively small; when the products are not critical or safety oriented, and design is relatively simple with relatively minimal documentation necessary; when agile-experts are continuously available in a critical mass; and in environments where the culture is one in which people thrive on chaos and are comfortable with several degrees of freedom.

1.29. Object-oriented analysis and design (OOAD) consists of methodologies and techniques based on objects (which combines data and processes) with activities (methods) rather than the traditional data description (data analysis) separate from processes (programming).

Answers to Problems and Exercises

1.30. The importance of using a systems analysis and design methodology is that a systematic step-by-step approach is taken that, if done correctly, results in a system that has fewer errors and is used with confidence by the users of the system. If shortcuts are taken for quick and easy development there is a greater chance for errors, as well as a system that does not meet user needs. The value of using an engineered approach results in a system that meets user needs while operating the way it was intended and built.

1.31. The similarities between the two figures are that they both contain the five phases of the SDLC. Figure 1.2, though, reveals a circular life cycle in which the useful life of one system leads to the beginning of another project that will lead to an improved or new system. Figure 1.3 is looked at as more of a spiral, in which phases are constantly cycling through at different levels of detail based on a Go/No Go axis where a decision is made to go through another cycle.

1.32. While figure 1.2 is circular in design, figure 1.4 is simply a vertical listing of steps that seem to indicate it is a one time through methodology that may not necessarily be the case. Many of the same SDLC techniques and tools are used. Many different companies have their own SDLC steps but in the end they are the same with some breaking them down into smaller sections to meet their own needs. For example, the U.S. Department of Justice's SDLC would break down as follows: 1) Planning: Initiation, System Concept Development, Planning; 2) Analysis: Requirements Analysis; 3) Design: Design; 4) Implementation: Development, Integration and Test, Implementation; 5) Maintenance: Operation and Maintenance, Disposition.

1.33. Figure 1.2 reveals a circular life cycle in which the useful life of one system leads to the beginning of another project that will lead to an improved or new system. Figure 1.9 has the same SDLC with one major exception. Figure 1.9 reveals an iterative relationship between the Analysis and Design phases that allows a design to be tested and revised until correct between the analysis and design steps. This design overcomes one of the main problems of the SDLC in that it does not treat these two phases to be separate and complete.

1.34. Object-Oriented Analysis and Design (OOAD) differs from the SDLC in that this methodology is based on objects rather than data or processes. The objects (a structure that combines both attributes and methods) is an abstraction of a real-world thing in which data and processes are placed together to model the structure and behavior of the real-world object. Putting the data and processes together in one place recognizes the fact that there are a limited number of operations for any given data structure. The main goal is to make system elements more reusable, thus improving system quality and the productivity of systems analysis and design. The four phases of the Rational Unified Process (RUP) are inception, elaboration, construction, and transition are not presented as a cycle because the objects are not part of an iterative design but carefully developed and tested together with both the data and process.

Guidelines for Using the Field Exercises

1.35. It should not be too difficult for students to list the various "systems" for an organization, perhaps for their university. These may include standard computer-based information systems, such as a transaction processing system for recording point of purchase sales or for registering for a course. They should also include systems that are not computer-based, such as a physical filing system for receipts or for transcripts. Instruct students to place each of their "systems" on their diagram and show how they (should) interact. Discuss the diagrams with students and have them evaluate whether or not they think that the systems interact well with one another. This enables students to determine whether the systems are well integrated (be sure that students have a clear idea on what it means for systems to be integrated). If their diagrams do not reflect well integrated systems ask them to draw a new "proposed" diagram in which they would interact in a better fashion. This is useful to get them thinking in terms of as-is and to-be systems.

1.36. Urge students to use their imaginations. You might have them imagine what would happen to the system design if one or more of the steps of the SDLC were ignored. For example, have students imagine what would happen if the planning phase was ignored. They might imagine elegant, costly systems that do not solve the right problem and, as a result, are not used. Alternatively, they might imagine a system where a database is kept redundantly in several different locations, or where information is re-keyed into one part of the system while it already exists in another format in another part of the system. They might describe a system that is lost completely because no proper backup and recovery procedures exist. The useful part of this exercise is that no matter what disasters or problems they imagine, they have probably already happened in one setting or another.

1.37. This is a useful exercise, particularly for beginning information systems students. This exercise enables students to see how information systems are used throughout an organization. Frequently, in smaller organizations, information systems development is more informal, and the various information systems roles are played by one or a small number of people. It is interesting to see how people in smaller organizations find creative ways to develop and implement technology on a limited budget and with a limited information systems staff. It is also useful to discuss how smaller organizations can integrate with systems outside the organization. For example, investigating how organizations, large and small, connect to credit card companies my provide students with a relevant task. Many instructors also find this is a way to inject service learning into this course as there are many well-deserving organizations out there that cannot afford systems consulting, but are desperately in need of systems assistance. Many of these projects could be the foundation for a course-long project that mirrors the concepts being taught in the course.

1.38. This is a great exercise, especially if the instructor has several local contacts from which students may choose. Otherwise students may find this exercise very difficult as often from the "outside" it is hard to know what methodologies and tools organizations are currently using. This would not normally be information found in the public domain. One approach to this exercise might be to have students find the local distributors for the major CASE products (Oracle, IBM/Rational and others), this might even be possible from a Web search so that students would have better knowledge of which organizations to contact. Once that contact is made, students could interview and then return to the class with the information and present the advantages and disadvantages that they have discovered to the class. Another important element would be where (in what SDLC phase) students found CASE was used most often and even which specific CASE tool was most often used. Care must be taken in an exercise of this nature

that the class doesn't generalize from a very small sample that CASE tools are always good and helpful or that they are always more challenging than they are worth.

1.39. Vendors are typically willing to provide this information, so your students should easily obtain the information they need for answering this question, with the possible exception to price but they should be able to find some relative numbers for comparisons. Nearly all software vendors have plans for and/or are working on future versions of their software. CASE tool vendors are no different. Since your students will obtain information about different CASE tools, it is useful for them to compare their answers. This would give the opportunity for presentations to the class or a team sharing exercise. This allows students to learn more about the competitive dynamics of the CASE tool market. You might suggest looking at a tool like LiquidPlanner. Encourage your students to visit the Web sites of various CASE tool vendors, including Oracle, Microsoft, Capterra, Borland, and Powersoft.

1.40. Encourage students to perform a search on the Web using search engines such as Google. A report or presentation as a deliverable from this exercise might be appropriate. Encourage students to consider how Agile methodologies differ from engineering oriented process. The nice thing about presentations to the class is that students have the opportunity to hone their communications skills and knowledge is shared amongst the class.

1.41. Journals are an effective teaching and learning tool. It is useful to collect these journals from time to time and provide direct feedback to each student, commenting on their experiences and answering their questions. You might also periodically have students share their journal comments and questions with the rest of the class and use this as fuel for class discussions. Even if students do not share actual comments, have them discuss the sources of their comments, such as newspaper articles, conversations with other faculty and students, advertisements, new topics they read in this textbook, or comments made by a parent. It might be interesting to note how students' thoughts on systems analysis and design change over the course of the semester. This exercise allows students to practice written communication and retrieval skills that will be necessary as they move out into the world of work and become project leaders and managers.