TT TEXTBOOK TESTBANKS

Solutions Manual for Modern Systems Analysis and Design 10th Valacich

SOLUTIONS MANUAL SAMPLE PREVIEW

PUBLISHER
Pearson

COPYRIGHT
2025

RESOURCE TYPE
Solutions Manual

ISBN
9780138180294

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 the Agile Methodologies, eXtreme Programming, and Scrum. The text compares and contrasts the new with more traditional methods in an effort to show both the advantages and limitations of these methods.

Chapter 1 introduces students to 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.

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 problems with the waterfall SDLC and explain the different approaches analysts, designers and developers have developed to improve the Systems Analysis and Design process.
6. Discuss Agile methodologies and eXtreme programming and how these compare to the traditional Systems Development Life Cycle (SDLC).

7. 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.
8. Compare and contrast the various development approaches introduced in Chapter 1 and depict how they all use an iterative approach.
9. 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 among 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 they 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 products, outputs or deliverables of each phase based on the in-text descriptions.
4. Ask students to compare Agile methodologies to traditional SDLC (see Table 1-3 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.
5. 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.
6. Discuss the benefits and organization design including roles for Scrum team members. Ask students to role play a typical sprint.

Answers to Key Terms

Suggested answers are provided below.

1. Analysis: The second phase of the SDLC, in which system requirements are studied and structured.

2. Application software: Computer software designed to support organizational functions or processes.
3. Design: The third phase of the SDLC, in which the description of the recommended solution is converted into logical and then physical system specifications.
4. Implementation: The fourth phase of the SDLC, in which the information system is coded, tested, installed, and supported in the organization.
5. Information systems analysis and design: The complex organizational process whereby computer-based information systems are developed and maintained.
6. Logical design: The part of the design phase of the SDLC in which all functional features of the system chosen for development in analysis are described independently of any computer platform.
7. Maintenance: The final phase of the SDLC, in which an information system is systematically repaired and improved.
8. Physical design: The part of the design phase of the SDLC in which the logical specifications of the system from logical design are transformed into technology-specific details from which all programming and system construction can be accomplished.
9. Planning: The first phase of the SDLC, in which an organization's total information system needs are identified, analyzed, prioritized, and arranged.
10. Systems analyst: The organizational role most responsible for the analysis and design of information systems.
11. Systems development life cycle (SDLC): (SDLC) The traditional methodology used to develop, maintain, and replace information systems.
12. Systems development methodology: A standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.

Answers to Review Questions

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

1.2. 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 at 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.

1.3. 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'.

1.4. 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 is that roles of system users or customers were 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.

1.5. 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.

1.6. 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.

1.7. Scrum originated in 1995 and is a popular methodology for agile development. Scrum was designed for speed development and multiple functional product releases. The primary unit is the Sprint, which will run for 2-4 weeks. Each Sprint is a complete project, which starts with an eight-hour planning meeting focusing on two questions, namely 1) what will need to be delivered by the end of the Sprint, and 2) how will the team accomplish that work? A Daily Standup 15-minute meeting during the Sprint to

evaluate progress made within the previous 24 hours. When the Sprint is completed it is followed by two additional meetings. The first one, the Sprint Review will last 4 hours, while the second one, the Sprint Retrospective, lasts three hours. The three primary artifacts in the Sprint process are: 1) the Product Backlog (an ordered list of everything included in the product), 2) the Sprint Backlog (a subset of the Product Backlog that lists only those items that are to be addressed in a particular Sprint), and 3) the Increment (which is the sum of the Product Backlog items completed during a sprint).

1.8. A study of Agile in practice has revealed three primary success factors. The first is a delivery strategy resulting in the continuous delivery of working software in short time scales. The second is that by following agile software engineering practices managers and programmers continually focus on technical excellence and simple design. The third is team capability of building projects around motivated individuals. It was also found that agile methods can lead to improved job satisfaction and productivity on the part of programmers.

1.9. 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 neither critical nor 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.

Answers to Problems and Exercises

1.10. The importance of using a systems analysis and design methodology is that a systematic step-by-step approach is taken which, 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.

1.11. 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.

1.12. While Figure 1-2 is circular in design, Figure 1-7 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. The traditional waterfall life cycle shown by Figure 1-7 has the property of locking users into requirements that had been previously determined, even though those requirements might have changed.

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 1-7 has the same major steps of the SDLC but by converting Figure 1-7 into a circle like Figure 1-2, we can interpret the project as being able to return to an earlier phase if necessary.

Guidelines for Using the Field Exercises

1.13. 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 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.

1.14. 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.

1.15. 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 might 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.

1.16. 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.

1.17. 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.