CHAPTER 14

CONSTRUCT, DELIVER, AND MAINTAIN SYSTEMS PROJECTS

This chapter examines the third through fifth stages of the SDLC—*in-house development*, *commercial package selection*, and *maintenance and support*. The amount of work required to develop a detailed design is extensive. But the quality of the resulting system reflects this effort.

The objectives of this chapter are:

- to be able to identify the sequence of events that constitute the in-house development phase of the SDLC;
- to be familiar with the tools used to improve the success of system construction and delivery activities, including prototyping, CASE tools, and the use of PERT and Gantt charts;
- to understand the distinction between the structured and object-oriented design approaches;
- to understand the use of multi-level DFDs in the design of business processes;
- to be familiar with the different types of system documentation and the purposes they serve;
- to understand the role of accountants in the construct and delivery of systems; and
- to understand the advantages and disadvantages of the commercial software option, and be able to discuss the decision-making process used to select commercial software.
I. In-House Systems Development

Looking broadly at the question of a new information system, there are two extremes: *in-house development*, wherein an organization develops the whole system itself, and *commercial system purchase* in which the entire system is bought from software vendors. Much of the “what” and “why” parts of the process is the same. It is the “how” that is different.

A. Tools for Improving System Development

Any creative process can be described as “loose.” And system development is no exception. There is no set of steps to follow precisely that will lead to the best possible system. No two organizations are the same, nor are their needs for information. And no two information systems are the same. So it should come as no surprise that, despite some guidelines and a well-accepted process, some systems projects are not successful. There are some projects that fail outright (up to 25%) and many more that are sub-optimal. This is in part due to the fact that the SDLC is not a linear process but is recursive.

The text discusses the three problems that can affect many systems projects:

- *poorly specified system requirements*,
- *ineffective development techniques*, and
- *lack of user involvement in the system project*.

Read the narrative carefully. These points are extremely important—especially the last. After problems occur is not the time to acknowledge these issues. In your career you will be involved in new systems development, perhaps sooner than you think.

Given the risks assumed when a new systems project is undertaken, it should not surprise you that techniques have developed to *automate the process of automating an information system!!!* Your text discusses three popular approaches. Read to understand the objectives of these methods.

1. Prototyping

*Prototyping* involves developing a “work-alike” model of a system so that users can evaluate it in use, to judge how it “feels.”

See Fig. 14-2, on page 702.
2. The Case Approach

*Computer-aided software engineering (CASE)* technology is kind of a sci-fi approach to system development—using the computer to design the computer systems. Read this material to get a feel for how sophisticated this approach is. The discussion of the different CASE models is good.

3. PERT Chart

The project evaluation and review technique (PERT) is used to represent the relationship among the key activities in a system project. **Fig. 14-4, on page 703** is a simple example.

4. Gantt Chart

This is a horizontal bar chart that relates the different activities to time elapsed. See **Fig. 14-5, on page 704** is an example.

II. Construct the System

One way to view the *construct* phase is to think of a blueprint of a house, etc. The basic characteristics are represented: how the rooms are positioned, how you get from one to another, sizes, etc.

Two basic approaches to conceptual design are currently in use: the *structured approach*, which is a top-down approach, and the *object-oriented approach*, which is a bottom-up or building-block method. The latter is being used more and more.

A. The Structured Design Approach

The *structured design* approach is represented in **Fig. 14-6, on page 706**. It starts with a big picture and breaks each step down further. This example shows the explosion of DFDs.

B. The Object-Oriented Approach

The *object-oriented design* approach is a building-block or bottom-up approach to system design. It is the trend of the future. Rather than reinventing the wheel, modules are developed and reused as needed. The example developed in the text is the building of an automobile. **Fig. 14-7 to 14-10, pages 707-08**.
C. System Design

The design process is not linear, but iterative—things learned in later steps will often require revisions to parts already “done.” The process is discussed in six stages:

1. data model the business process and design the conceptual views;
2. design normalized base tables;
3. design physical user views;
4. design the system process;
5. design system controls; and
6. perform a system design walkthrough.

Each stage is discussed in detail. Read and study carefully.

D. Data Modeling, Conceptual Views, and Normalized Tables

Data modeling is the process of specifying the data requirements of a business process. This is documented using entity relationship diagrams, which were introduced in Chapter 2. Once modeled, the data attributes that define an entity can be described—this is a conceptual view which is the basis for the database tables. Refer back to the discussion in Chapter 9 of normalized database tables and why normalization is an issue.

E. Design Physical User Views

Table 14-1, on page 711, gives some examples of outputs from the various sub-systems. Attention must be paid to the difference between discretionary and nondiscretionary reporting. Every desired report must be planned and the format, timing, frequency and elements (and ultimately the inputs and required processing) specified.

The attributes of desired output deserve close attention. You may wish to refer to your intermediate or advanced texts for discussion of the characteristics of accounting information as presented in Statement of Financial Accounting Concepts #2. There is often a tradeoff between various characteristics or attributes, and you should understand the implications of such tradeoffs.

Output can be either paper or electronic, or a combination of the two. There are two basic forms of input: hard-copy and electronic. There are many
factors to be considered in designing hard-copy: storage, handling, number of copies, size, format, etc. Read the discussion of form design carefully. Good forms help the input process, and bad forms can cause real problems. The use of zones and embedded instructions are very simple concepts, yet are very beneficial.

There are two ways to approach electronic input: from source documents and directly. Think of some of the situation you are in regarding input.

F. Designing the System Process

This section will be most informative to those of you who have had some programming experience. Each step that has been represented on a DFD must be decomposed into its most elementary parts. The example of a purchase system is given, and the accounts payable process is decomposed (see Figs. 14-15 to 14-17, on pages 717-719). Watch how the steps are broken down, and don’t get lost.

_Pseudocoding_ is a technique used to describe what must happen without using the programming language itself. It can aid understanding by non-programmers.

G. Designing System Controls

Although most of the controls will be designed as the parts of the system are assembled, this stage permits a review to be sure there are no voids in the control systems.

H. Perform a System Design Walkthrough

Before the final OK is given to implement the newly designed system, a _walkthrough_ is conducted. This is a process whereby independent systems professionals examine the processing “line-by-line” to evaluate the logic, etc. At this time, a thorough review of the system documentation to this point is also conducted. The detailed design report is quite thorough. Note what is included. If judged inadequate, the system undergoes additional design work. If all is well, the system moves on to implementation.

I. Program Application Software

Ideas are one thing, but eventually the actual computer code must be written. This section will give
you some ideas about computer languages if you have never taken a programming class. In particular, there is discussion of procedural languages, even-driven languages, and object-oriented languages.

Obviously, the work of programming a new system must be done by experienced and skilled programmers, not accountants. The discussion of language choice is important but will not be made by you. However, the importance of testing the software has implications for auditing.

J. Software Testing

As each piece of the system is completed, they must be thoroughly tested. The example in the text is simple but clear.

III. Deliver the System

A. Testing the Entire System

In no way should a newly developed system be implemented without thorough and extensive testing. Note the importance of saving the ext data!

B. Documenting the System

There are many types of documentation that must be completed during implementation. There are four groups of people for whom documentation is very important: designers and programmers, operators, end users, and accountants (auditors). Focus on the latter two in your reading. Note, however, the impact of internal control on what is and is not in the documentation for different groups. For example, operators do not need program code, flowcharts, etc. Also note the discussion of the different levels of users.

C. Converting the Databases

Converting the data in the old system to the new system can be very cumbersome. It is critical to successful system implementation. If the data is already in machine readable form, some of the conversion can be automated. Note in particular the three precautions.
D. Converting to the New System

Moving from the old system to the new can be a nerve wracking experience. Your book discusses three approaches: cold turkey cutover, phased cutover, and parallel operation. Although the latter is the safest, it involves doing everything twice for at least one, if not more, cycle. This is costly and stressful for personnel.

E. Postimplementation Review

After the new system is in place, the entire process is evaluated. The best time to assess the system development process is right after completion of a system project. Not only is the new system evaluated, so is the whole system development process. By carefully documenting what took place, better planning can occur the next time a system project is undertaken.

Two main areas of concern are presented: design adequacy and accuracy of time, cost, and benefit estimates. This information can keep the next project from making the same mistakes.

F. The Role of Accountants

Pay close attention to the three roles played by accountants. You will play some, or all, of these roles some day.

IV. Commercial Systems Packages

Many organizations choose to purchase commercial systems rather than try to develop their own.

A. Trends in Commercial Software

Your text discusses four trends in commercial systems:

- turnkey systems,
- backbone systems,
- vendor-supported systems, and
- ERP systems.

The differences are important. Focus on the advantages and disadvantages of commercial software. Both must be considered.
B. Choosing a Package

Your text outlines a four-step procedure to selecting commercial software:

- Needs analysis,
- Send out the request for proposals,
- Gathering facts, and
- Analyze the findings and make a final selection.

This is more elaborate than you might consider for small personal software decisions but can be of some benefit for you as well as your employer or client.

V. Maintenance and Support

Maintenance can involve updating commercial software and/or modifying in-house systems to keep up with user need. Neither is always a simply process. Documenting changes is crucial to this process. Your book discusses two issues that are extremely important: the need for user support (e.g., help desk, training, etc.) and knowledge management and group memory (collecting information and retaining it).

APPENDIX

Read this material carefully to understand the basic ideas. You may be involved in a new system project sooner than you expect. Output reporting alternative should pull together ideas you have met before.

Review Questions for Chapter 14: 1-47

Discussion Questions for Chapter 14: 1-21