

Systems Development

Chapter Outline

- Introduction, 773
- Building Information Systems, 776
 - Custom Programming*, 777
 - Outsourcing and Contract Programmers*, 778
 - Assemble Applications from Components*, 779
 - Purchase an External Solution*, 780
- Computer Programming, 781
 - Programming Logic*, 781
 - Events*, 783
 - Object-Oriented Programming*, 785
 - From Programming to Development*, 787
- Systems Development Life Cycle, 788
 - Introduction to SDLC*, 789
 - Feasibility and Planning*, 789
 - Systems Analysis*, 791
 - Systems Design*, 792
 - Systems Implementation*, 793
 - Maintenance*, 796
 - Evaluation*, 797
 - Strengths and Weaknesses of SDLC*, 798
- Alternatives to SDLC, 799
 - Prototyping or Iterative Development*, 800
 - Extreme Programming and Agile Development*, 800
 - Developing Systems Requires Teamwork: JAD and RAD*, 802
 - Communication*, 806
 - Object-Oriented Design*, 806
 - Open Source Development*, 807
 - End-User Development*, 808
 - Development Summary*, 809
- Process Analysis, 810
 - Input, Process, Output*, 810
 - Divide and Conquer*, 812
 - Goals and Objectives*, 813
 - Diagramming Systems*, 814
 - Summary: How Do You Create a DFD?*, 818
- Object-Oriented Design, 818
- Distributed Services, 819
- Cloud Computing, 820
- Summary, 821
- Key Words, 823
- Web Site References, 823
- Review Questions, 823
- Exercises, 824
- Additional Reading, 828
- Cases: Government Agencies, 829

What You Will Learn in This Chapter

- How do you create the software tools needed for your organization?
- What main options exist for building information systems?
- What are computer programs?
- How do you control a major development project?
- Is SDLC always the best approach? What other methodologies could be used?
- How do you analyze and annotate a process-based system?
- How is object-oriented design different from process design?
- Can software be located in multiple places?
- How does cloud computing change software development?

Federal Aviation Administration

Why is it so difficult to develop software? Developing software for complex systems is a difficult task. Even bringing in outside contractors has not solved all of the problems for the FAA. Cost overruns and missed schedules have happened so many times in FAA (and other government) projects that completing a project on time is the rare exception.

The FAA is charged with controlling civilian and military uses of U.S. airspace. The FAA is also responsible for modernizing the airways, installing radar, and training air traffic controllers. Probably their best-known function is control over commercial flights and routes to maintain safety and efficiency. With 50,000 flights a day among 300 major airports, the FAA has a huge task.

The FAA has a computer system to help it control the thousands of daily flights. However, the system was created in the early 1960s. It has been patched and upgraded, but most of the hardware and software are based on decades-old technology. On several occasions, the FAA attempted to upgrade the facilities, but complications have forced the agency back to the old technology.

The FAA systems are particularly difficult since they entail high-risk operations that have to be performed accurately. Also, the technology is relatively unique and development often requires state-of-the-art skills in areas that are rarely taught or researched in schools. Also, large-scale projects are always difficult to forecast, because any number of things can go wrong.

Introduction

How do you create the software tools needed for your organization? Today, most companies have chosen not to develop software. In most cases, it is simpler to buy the tools from existing companies. Still, tools often need to be customized and sometimes small applications can make a big difference in how your company operates. As a business manager, you will probably not become a programmer or systems developer. On the other hand, at some point, you will have to interact with developers to ensure that you get the systems you need. You need to be able to communicate with the developers, and you need to understand their constraints. Furthermore, as a manager, you need to understand the costs and management issues facing MIS departments. You might have to choose among development methodologies, so you need to know the options and their strengths and weaknesses.

As shown in Figure 12.1, creating software systems is a difficult task, similar to creating highly-complex buildings. The tools have gotten better over time, but the tasks are much harder. Several cases and stories exist of failed projects—some costing millions of dollars ended up being discarded. Some studies have reported that half or more of software development projects are over budget and behind schedule. However, these numbers are merely estimates because most schedules are best guesses. Building anything entails guesswork on schedules. If you are creating something you have built a hundred times before, you will have a pretty good idea of how long it will take. But, software generally involves creating totally new systems—often with new hardware and new tools. This process is more like the challenges faced by Boeing



Figure 12.1

It is not easy to create information systems to support business needs (strategy, tactics, and operations). Three basic development techniques are: systems development life cycle, prototyping, and end-user development. As a manager, you will participate in each of these methods. You will sometimes have to choose which method to use.

in designing and building the new 787 model—which fell several years behind schedule. Boeing is certainly one of the best companies at designing and building airplanes, yet the challenges faced in using new materials and new construction techniques caused many problems and delays. With large complex projects, it simply is impossible to predict all possible delays.

Small projects that can be built by one person are relatively easy. The challenge lies with large projects that involve many users and many programmers. These systems need to be split into smaller pieces—where the pieces are assigned to teams and eventually to individual programmers. But, splitting a project means that everyone has to work together to share their designs and progress. The pieces have to fit back together and still meet the overall objectives. The system needs a solid plan and control mechanisms to ensure everything works together and stays on schedule.

The essence of managing software development comes down to coordinating dozens or hundreds of programmers and several methodologies have been created to handle the common tasks. The most formal approach is known as the **systems development life cycle (SDLC)**. Its structure and concept were borrowed from engineering and construction of large physical items such as bridges and dams. Large organizations that develop several systems use this method to coordinate the teams, evaluate progress, and ensure quality development. Most organizations

Trends

Internally, computer processors have limited capabilities. The processor has a set of a few hundred internal instructions that it knows how to perform, such as moving a number to a new place in memory or adding two numbers together. Initially, all computer programs were written at this low level; but writing programs at this level is difficult and time consuming. Over time, two major innovations were created to reduce the difficulty of programming at these low levels: (1) higher-level languages were created that handle many details automatically for the programmer, and (2) common algorithms used by many applications were created and sold as operating systems. These advances enable programmers to focus on the applications instead of machine-specific details.

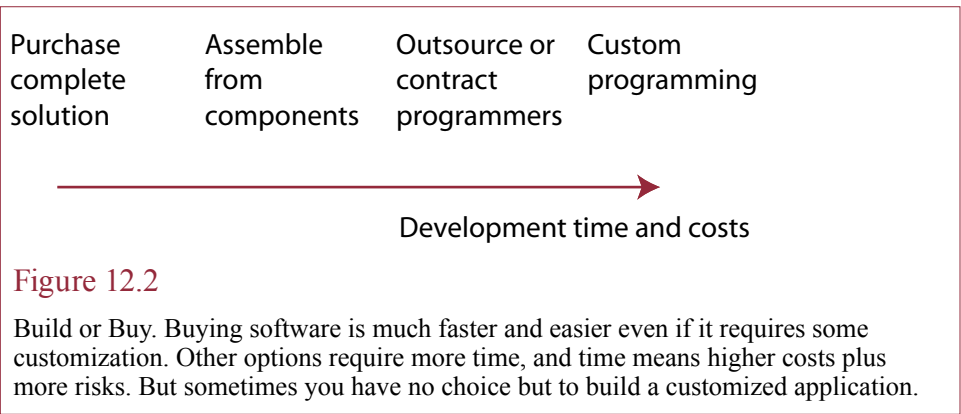
Despite many advances, writing programs is still complex and time consuming. Large-scale applications require the teamwork of millions of hours of programmer time and cost hundreds of millions of dollars to build and maintain. These high, fixed development costs underlie the growth of the commercial software industry. By making the software relatively generic and selling it to hundreds or thousands of firms, the development costs are spread over a wider group.

Increasingly, firms are moving away from custom-written software. They are purchasing packages and hiring outside programmers to develop many components. In these situations, design issues generally consist of choosing and customizing the software to meet the individual needs of the organization.

have created their own versions of SDLC. Any major company that uses SDLC also has a manual that is several inches thick (or comparable online documentation) that lays out the rules that MIS designers have to follow. Although these details vary from firm to firm, all of the methods have a common foundation. The goal is to build a system by analyzing the business processes and breaking the problem into smaller, more manageable pieces.

Improvements in technology also improve the development process. The powerful features of commercial software make it easier to build new applications. Programmers and designers can work with larger, more powerful objects. For example, instead of programming each line in COBOL, a report can be created in a few minutes using a database management system or a spreadsheet. **Prototyping** is a design technique that takes advantage of these new tools. The main objective of prototyping is to create a working version of the system as quickly as possible, even if some components are not included in the early versions. A third method of creating systems, **end-user development**, relies on users to create their own systems. This method typically uses advanced software (such as spreadsheets and database management systems) and requires users who have some computer skills.

It is important to be careful when you implement any new system. Case studies show that major problems have arisen during implementation of systems. In fact, some organizations have experienced so many problems that they will deliberately stick with older, less useful systems just to avoid the problems that occur during implementation. Although changes can cause problems, there are ways to deal with them during implementation.

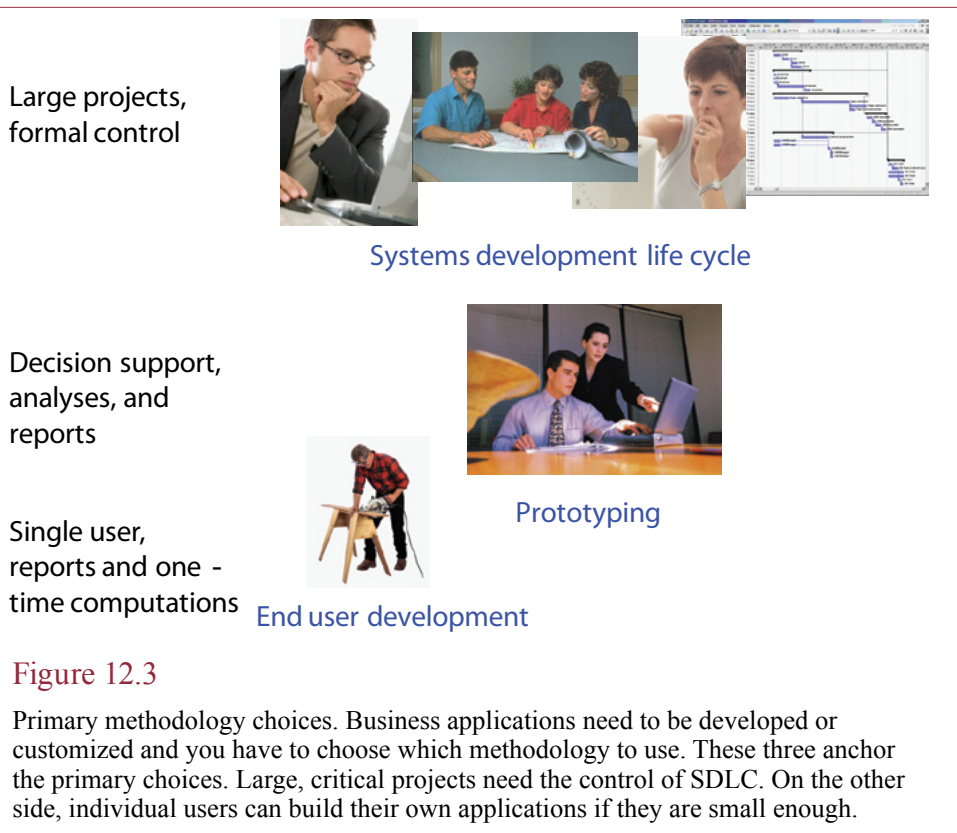


There have been some spectacular failures in the development of computer systems. Projects always seem to be over budget and late. Worse, systems are sometimes developed and never used because they did not solve the right problems or they are impossible to use. Several design methods have been created to help prevent these problems. All methods have advantages and drawbacks. As a result, they tend to be suitable for different types of problems. As a manager, you need to be aware of these different tools and their strengths so you can ensure the right methodology is used for your projects. Sure, you could call in EDS, Accenture, or IBM to work on all of your projects—but it would be absurdly expensive to use these formal methods for tiny projects. You know not to call Boeing to help you build a paper airplane, but how do you know when you need help turning your hundreds of spreadsheets into a Web-based application?

Figure 12.2 shows one of the initial issues you will face when looking for new systems. Buying a system is the easiest, fastest method of getting a new application that works. It might have to be customized to meet the needs of the company, but customizing existing software is easier than starting from scratch. If the exact tool you want is not available, you might be able to build something close using other components. For instance, a Web site could be created by adding a shopping cart, chat room, and payment processing components. If you need a more specific component, you could hire a contract programmer to write it. Creating small, clearly-defined components is usually straightforward and can be done without huge management costs. For truly unique systems, it is still necessary to hire programmers and developers and build most of the elements from scratch. This type of development requires the most oversight and control.

Building Information Systems

What main options exist for building information systems? Figure 12.3 illustrates the three primary methods of creating or customizing a business application that you are likely to encounter as a manager. When you need to create or customize an application, you ultimately have to choose some variation of these three. Many alternatives exist, but these three represent anchors. SDLC tools are used for large, critical projects involving many developers because they foster control and repeatability. On the other side of the spectrum, small projects such as reports, queries, charts, and data analysis can be handled completely by end users—with the appropriate training and experience. As long as you, the user, has the tools and business knowledge, it will often be faster for you to develop



and customize applications to meet your own needs. If you lack the skills to build the application yourself but the project is not too complex, a developer with the skills and tools can work with you to design and build the application.

Another way to look at the problem of software development is to recognize that ultimately developers have to write detailed program code. If your needs are similar to those of other businesses, companies might have already created a commercial application that you can use or customize. If you can buy existing software or even components, the cost is substantially lower, and you reduce risk by using code that has already been tested.

Custom Programming

Ultimately, all applications are created by teams of programmers writing detailed code. Writing your own custom program gives you complete control over the application. You can include any features, build in special routines unique to your company, and integrate the data with your existing systems. The problem with creating your own code is that programming is difficult, time consuming, hard to control, and expensive. Even when the application is completed, you will still need groups of programmers to fix problems, add new features, and develop future versions of the software. Modern development tools make it easier to write programs today, but every application still requires intense development efforts, and the tools never seem to provide all of the features you need.

Reality Bytes: System Development Methodologies

Accenture is the largest worldwide consulting firm in management information systems. It conducts major installations using a proprietary methodology called Method/1. Method/1 uses four phases in the development process: plan, design, implement, and maintain.

McKinsey and Co., a strategic consulting firm, examines organizations with a copyrighted “Seven S” model. The Seven S’s are structure, systems, style, staff, skills, strategy, and shared values.

Electronic Data Systems (EDS), started by Ross Perot, purchased by GM, and now independent, is the largest outsourcing company. It also develops systems using a traditional SDLC methodology. The detailed methodology has thousands of individual steps spelled out on separate pages that must be signed off at each step.

Rational Rose, a tool to support object-oriented development, was designed by the leading OO gurus. It is a graphical tool designed to show the object details and their relationships. It supports reverse engineering, by reading existing code and converting it into the corresponding diagrams. The Rose tool can also generate the final code from the diagrams.

Microsoft, as a commercial software vendor developing highly complex systems used by millions of people, has developed its own methodologies for quickly creating code. The company has been a proponent of Rapid Application Development to reduce the time it takes to complete large projects. The methodology is designed to segment the code so that hundreds of programmers can work on it simultaneously. The methodology also relies on creating code that can be modified later, for improvements and patches.

With *Open Source*, led by Richard Stallman and the GNU project, thousands of programmers around the globe are building complex projects in a loosely-knit organization. Generally, one person is a project leader responsible for setting strategies and resolving disputes. All of the source code is publicly available to anyone (hence the name). Interested programmers suggest improvements and submit the modifications. Some complex systems have been built this way with minimal central control.

Quality and security are two critical issues with any type of development. If you build a custom application, you must leave sufficient time to build in quality tests and to correctly build security controls. Employees also have to be trained in these two areas, and you generally need specialized managers to oversee each of these two areas.

Outsourcing and Contract Programmers

One of the other problems with creating your own software is that you generally have to hire many programmers at some point in the development. But when the development is finished, you will rarely need all of these programmers. So, either you find new tasks for them or you have to release them to reduce your costs. But hiring and firing workers for a single project is frowned upon. Consequently, many firms use contract programmers, or even outsourcing, to handle system development.

With contract programming, you negotiate with a company to provide specialists for a given period of time. When their work on the project is complete, they move on to another job. The process saves you the problems of hiring and firing

Reality Bytes: It Doesn't Get Easier

Developing software is hard. The bigger the project, the harder it is to manage the team. Engrave these rules in stone, developers have faced them for decades, and still the process does not get easier. Fred Brooks (*The Mythical Man Month*) described the problems in detail regarding a 1960s IBM project. Scott Rosenberg (*Dreaming in Code*, 2007) explains how the problems still exist. He was project manager on an open source project to create a new personal information manager called Chandler. Two dozen programmers struggled on the project for over three years. The project is still unfinished (0.7) and searching for more developers. In an interview, Rosenberg points out two major insights into the development process. "If we try to design a system that is fully featured and complex and does everything everyone wants it to, we'll never have any system." "Break things up into small bites. Any opportunity you have to do that should be seized." He also emphasizes that it is difficult, expensive, and time-consuming to develop new software. For example, "even at Microsoft prices it's cheaper to use Word than to write a new word processor." New software is developed when people find new tools and need better ways to work with data. But developing software is hard. Start with a reasonable goal and get something out the door. If it works, improve it.

Adapted from Edward Cone, "Scott Rosenberg: What Makes Software So Hard," *CIO Insight*, January 5, 2007.

large numbers of employees. On the other hand, contractor salaries are usually higher than traditional employees. More important, several lawsuits have made it critical that you clarify the exact role of contractors—otherwise they can sue to be classified as regular employees to gain additional benefits such as stock options.

Outsourcing goes a step further than contract programming. When you outsource project development, you transfer most responsibility to the outside firm. Typically, you negotiate a development price, provide detailed specifications, and the outsourcer hires workers and develops the system. A huge variety of outsourcing arrangements are available, including situations where the outsourcers run your entire MIS department or just your servers or networks, or handle PC maintenance.

The primary advantage of outsourcing is that the external company takes responsibility for managing the process and producing the application. You still have the responsibility to clearly define exactly how the application should work, but the outsourcer bears more of the risk—particularly with fixed-fee contracts. The one thing you want to avoid with contractors and outsourcers is uncontrolled hourly fees.

Assemble Applications from Components

A good way to reduce development time and costs is to buy portions of the system from other companies. Even if you need a custom solution, you can purchase a variety of software components that handle many of the difficult tasks for you. Components are a powerful feature of modern operating systems. They are blocks of code that are integrated into custom applications. For instance, you could purchase a security control to handle encryption on a Web site. Whenever your application needs to encrypt or decrypt some data, it simply calls the component's

methods. Similarly, if you need to process a credit card application, you can install a component (or link to a Web service) that handles everything for you. Thousands of useful components are available for a few hundred dollars each or less. You simply install the component on your server and your programmers can begin using the functions within their code. This approach relies on the capabilities of **commercial off-the-shelf (COTS) software**. As the number and quality of software packages have increased, it has become easier to build a system based on COTS.

Increasingly, tasks currently performed by components are being offered as services over the Internet. The same principles will apply, but the external company will maintain the application, install upgrades, and add new features. In addition, new services will become available. For example, services provide instant exchange rate conversions so that you can list prices in any currency. The conversion rates will always be current with no effort on your part.

Components have many substantial advantages and only minor drawbacks (primarily the price hassles with upgrades). They can significantly reduce development time and provide powerful features that are beyond the capabilities of many staff programmers. In fact, many outsource specialists develop their own collection of components to use in developing custom solutions. By integrating commonly used features, they can build new applications faster with fewer errors.

Purchase an External Solution

Taking the concept of components and outsourcing a step further, many commercial software companies sell prepackaged applications. Some are *turnkey* systems where you simply load your data, select a few preferences, and the system runs (much like buying an automobile, you turn the key and no assembly is needed). Other applications require detailed customization. The ERP packages (such as SAP) are classic examples. The system handles all of the basic operational data of the firm, including generation of financial reports. You purchase the software from the vendor and install it on top of a database management system. You still have to set up your accounts and some custom details for reports. The application can then be used by your company to track all financial and manufacturing data and produce standardized reports.

On the other hand, you can also customize most of the features. If you need unique manufacturing reports, you can write code to generate them. The degree of customization often depends on the attitude of management. The drawback to extensive customization is that it requires specially trained programmers and delays the entire project. Moreover, when the DBMS vendor or the ERP vendor upgrades the underlying software, you may have to rewrite all of your custom programs.

Prewritten packages can have high price tags (SAP costs can easily run into millions of dollars). But it could take millions of hours of programmer time to create a custom system with the same functionality.

In general, it is almost always preferable to buy solutions, but keep a close eye on prices. The commercial software essentially spreads the development costs across thousands of firms. Unless you have a truly unique application and are willing to pay a staff of top-notch programmers, it is better to share the development costs. And if you do have a radically different application, you should consider packaging it and selling it to other firms to reduce your costs.

Sequential execution: Statements are executed in order.
Parallel execution: Groups of code are executed at the same time.
Variables: Containers to hold data
Computations
Conditions: If – Then – Else
Loops: While – End
Subroutines and Functions: Break code into manageable pieces.
Input/Output: Transferring data

Objects: Code pieces purchased or created to do specific tasks.

Figure 12.4

Programming structures. Computer programs are developed by using these basic concepts. You can purchase prewritten objects and functions to handle many common tasks. But, you have to understand the objects and learn how to use them.

Computer Programming

What are computer programs? To begin to understand the development process, it helps if you know a little bit about how programs are written. The goal of this section is to help you understand the level that programmers deal with and to help you read the initial products. Programmers often write pseudocode or outlines that you need to be able to read to ensure the program will work correctly.

Of course, it takes several books, considerable practice, and good math skills, to become a programmer. But, the foundations of programming are relatively easy to understand. You should check out the Toolbox on programming to see how managers can write a few lines of code to help solve some relatively complex tasks. Figure 12.4 shows the fundamental concepts of programming. Essentially, processors execute one line of code at a time and follow your instructions. Parallel processing is more difficult because you can have several sections of code executing at the same time, which means you have to worry about which one might finish first; but it leads to faster programs. Variables, computations, conditions, loops, subroutines, and input/output are common features of almost every language. Once you learn the **program logic** of these commands, you can write code in almost any language. Computer languages also possess a **syntax** that dictates exactly how you have to type in the commands, including spelling (case-sensitive), commas, dots, and other annotations. Development tools (such as Microsoft Visual Studio and the open source Eclipse) make it easier to enter code correctly by prompting with syntax options and verifying the items as you type them.

Programming Logic

Figure 12.5 shows a tiny sample program. Instead of following the syntax of a particular language, it uses pseudocode to illustrate the logic. Several additional statements would be needed to turn this code into an application that runs. On the other hand, almost no one would ever write this code because you can use SQL to accomplish the task easier and faster. But, its simplicity makes it a good starting point. Pay attention to the role of the Total variable as an accumulator. It represents one location in memory that acts as a container. Initially, zero is placed in that container. Each time a row of data is read, the number currently stored in the

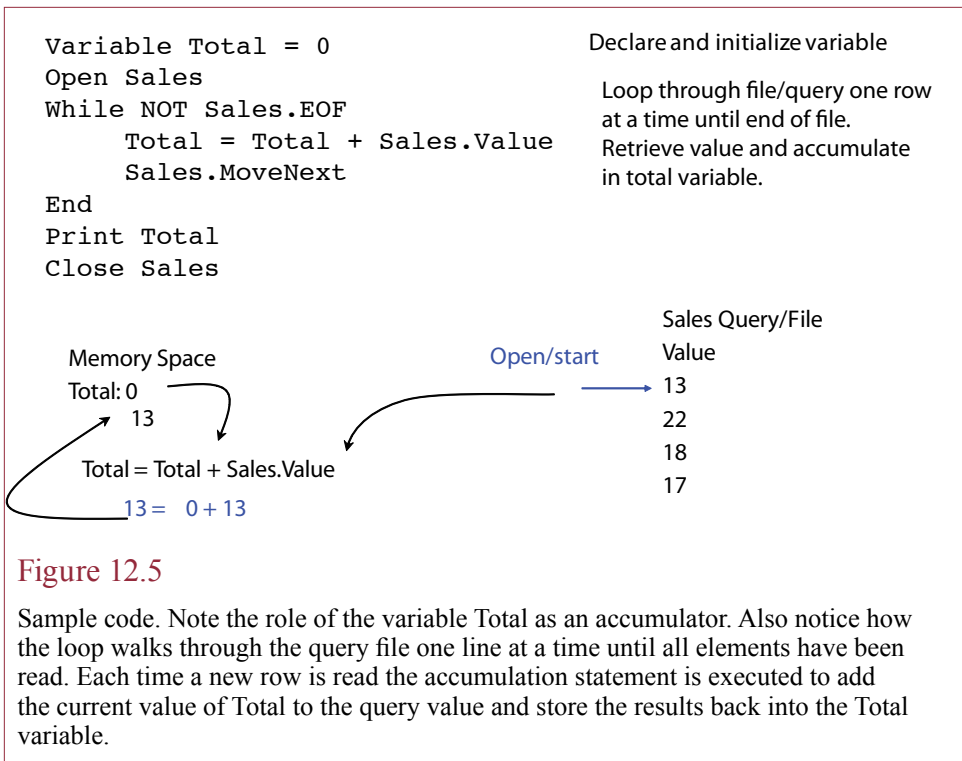


Figure 12.5

Sample code. Note the role of the variable *Total* as an accumulator. Also notice how the loop walks through the query file one line at a time until all elements have been read. Each time a new row is read the accumulation statement is executed to add the current value of *Total* to the query value and store the results back into the *Total* variable.

Total location is retrieved and added to the value from the query. This computation is performed within the computer processor itself. The result is transferred back to memory. The loop causes the processor to retrieve each row from the query and execute the accumulation statement until the end of file is reached. The program code itself is also stored in memory while it is being executed. Loops and conditions are common elements of any program. A conditional statement consists of an IF statement, such as *if (Total < 0)*, and a set of statements to execute only if the condition is true. Many programs also contain *else* statements that are executed only if the condition is false.

Writing code is partly science and partly art. It takes practice to learn how to translate what you want into detailed code statements. But, writing code can be fun and creative. You get to tell the computer exactly what you want it to do, and you get to create something that people can see and use. Programmers also have to be detail-oriented. Small errors in logic or typing can be difficult to find and correct. This combination of creativity, logic, and detail-orientation is difficult to find; which partly explains why good programmers receive relatively high salaries.

Figure 12.6 lists a few of the programming languages in use today. Many of the languages have a similar syntax—loosely based on the earlier C language. But subtle syntax and critical structural differences exist among the languages. A key difference exists with C and C++ because they are both compiled down to assembly code which provides detailed control over the computer and has high performance. Many of the other languages run on an intermediate level which handles data and interfaces differently. It is easier to write code in these languages (such as Java and C#), but performance can be an issue for some types of problems. Other languages (notably Javascript) are dynamic or script languages which are often

Language	Main Purpose or Context
Java	General purpose. Designed to run on servers and clients.
C#, VB, ASP .NET	Microsoft used for Web servers and applications. Managed code.
Javascript, HTML5	Script language for Web page interactivity.
C++, C	Compiled, powerful, lower-level languages often used for systems and tools or where speed is critical.
PHP, PERL, Python, Ruby	Web server scripting/dynamic languages often used on Linux/UNIX or Apache Web servers.
Objective-C	Apple's language for iPhone/iPad applications.
Flash (Adobe) Silverlight (Microsoft)	Special purpose Web add-ins with code to control interactivity. Silverlight uses C# or VB.
COBOL	Older business applications, and SAP.
FORTRAN	Older scientific programming applications, but check out F#.

Figure 12.6

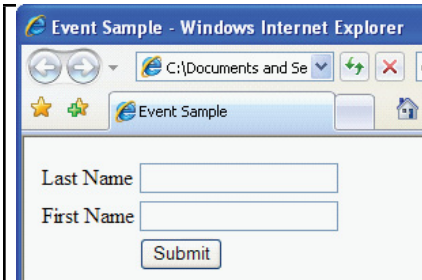
Some programming languages. Religious wars have arisen among geeks over which language is better. Many of the languages have a similar syntax, but the framework and structures can be radically different.

parsed from text only when the code is executed (run time) and can run even more slowly.

Programming requires learning more than just the syntax and structure of a given language. Each program runs within a framework or environment that can limit the actions available and also provide detailed tools and options. For instance, the Windows operating system provides detailed controls for interacting with the screen, the user, and other devices. Writing code for a Web server uses a different set of tools, libraries, and events.

Events

Today, understanding programming logic and syntax is only the first step in learning to program. Most programming environments are **event-driven** systems where programmers create code function that are executed when some event is triggered. You are familiar with these systems as a user, but you might not have realized what was happening. Most of the user interfaces today are based on graphical screens and forms, including Web pages. Each of these environments defines dozens or hundreds of events. Programmers can write code that attaches to an event and then action is taken when that event occurs. As a simple example, Figure 12.7 shows a partial Web form and lists the events that can be activated by the Submit button. A programmer can write Javascript code for any of those events that will



onactivate onafterupdate onbeforeupdate onbeforecopy onbeforecut onbeforedeactivate onbeforeeditfocus onbeforepaste onbeforeupdate onblur onclick oncontextmenu oncontrolselect oncopy oncut ondatabinding ondblclick ondeactivate ondisposed	ondrag ondragend ondragenter ondragleave ondragover ondrop onerrorupdate onfilterchange onfocus onfocusin onfocusout onhelp oninit onkeydown onkeyup onload onlosecapture onmousedown onmouseenter	onmouseleave onmousemove onmouseout onmouseover onmousewheel onmove onmoveend onmovestart onpaste onprender onpropertychange onreadystatechange onresize onresizeend onresizestart onselectstart onserverclick onunload
---	--	--

Figure 12.7

HTML Javascript events for the Submit button. Any event can cause code to execute. Having more events gives programs more flexibility, but rarely are more than three or four used for a specific problem.

be executed when the event is fired. If you think 56 events is too many, go online and check to see how many exist now. Adding events creates more flexibility and granularity in responding to subtle changes, so new events are periodically added to the Web standards.

In most situations, as the programmer, you have to identify which event most closely matches an action that you want the code to take. Rarely will you need more than one or two events for a given object, and the onclick event is popular. In this example, the onclick event is fired whenever the user clicks the Submit button. However, it is often better to use the onsubmit event associated with the form object instead. The onsubmit event is triggered regardless of how the form is submitted (via a click or via some other lines of code). The distinction is subtle, but one of the tricky aspects of programming is that you need to completely understand all of these subtle differences—for hundreds or thousands of events. To make it easier to learn to program and to reduce mistakes, most programmers and companies develop a set of **best practices** that spell out the best way to handle typical problems. Good programming books list and follow these best practices.

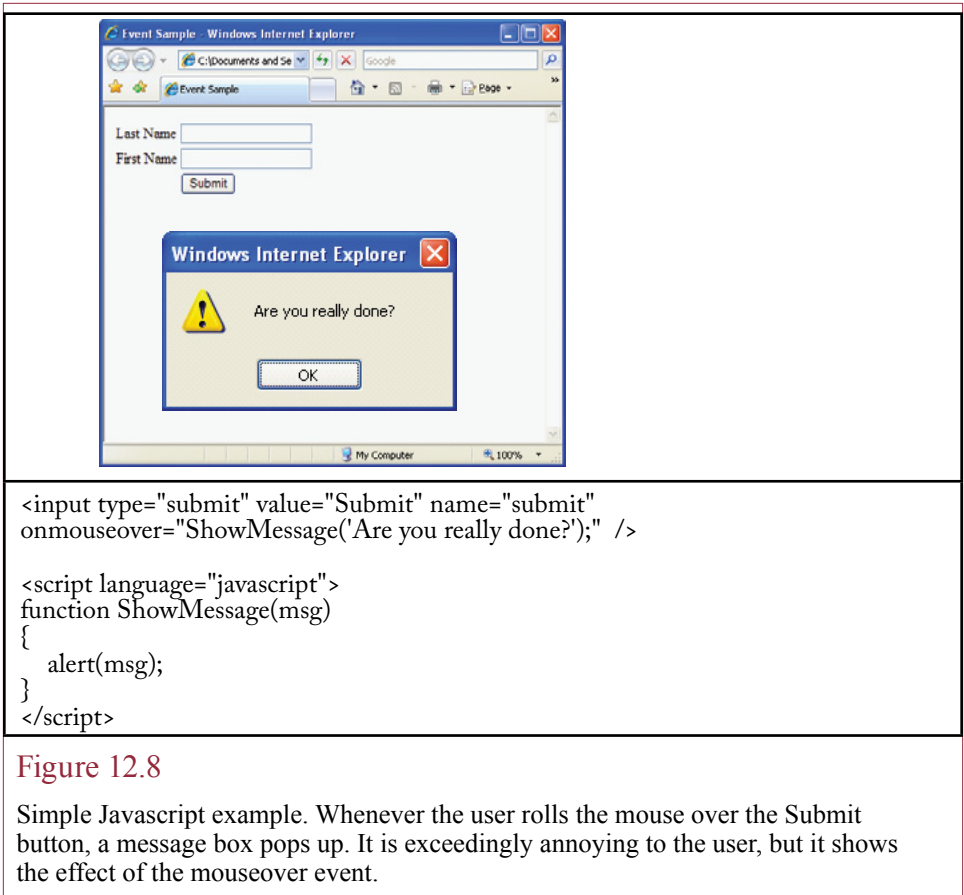


Figure 12.8

Simple Javascript example. Whenever the user rolls the mouse over the Submit button, a message box pops up. It is exceedingly annoying to the user, but it shows the effect of the mouseover event.

Figure 12.8 shows a couple of simple lines of code that are executed when a user rolls the mouse cursor over the Submit button. You would never write this code on a real Web page because alerts are exceedingly annoying. However, it is an easy way to verify that the code is executing properly. With a little thought, you could write some conditions for the function that are useful—such as checking to see if the user entered data in all of the boxes. The art of programming revolves around building applications that solve problems and are easy to use.

Object-Oriented Programming

Object-oriented programming (OOP) evolved in the 1990s as a way to make it easier and faster to create new applications. With this approach, the focus is on defining objects that have specific properties (data) and methods (code). All of the common applications you use; including graphical operating systems, personal productivity tools, and Web browsers; were built using OO techniques. The point of this approach is that the internal objects created to build these tools are available for you to use in other programs. For example, you can write a new business program that uses a spreadsheet object and its internal tools. **Reusability** of objects is an important feature of the OOP approach.

Business programmers can use similar techniques to create their own objects. Objects are defined by a set of properties (or attributes). The properties define the object. They also represent data that will be collected for each object. Consider

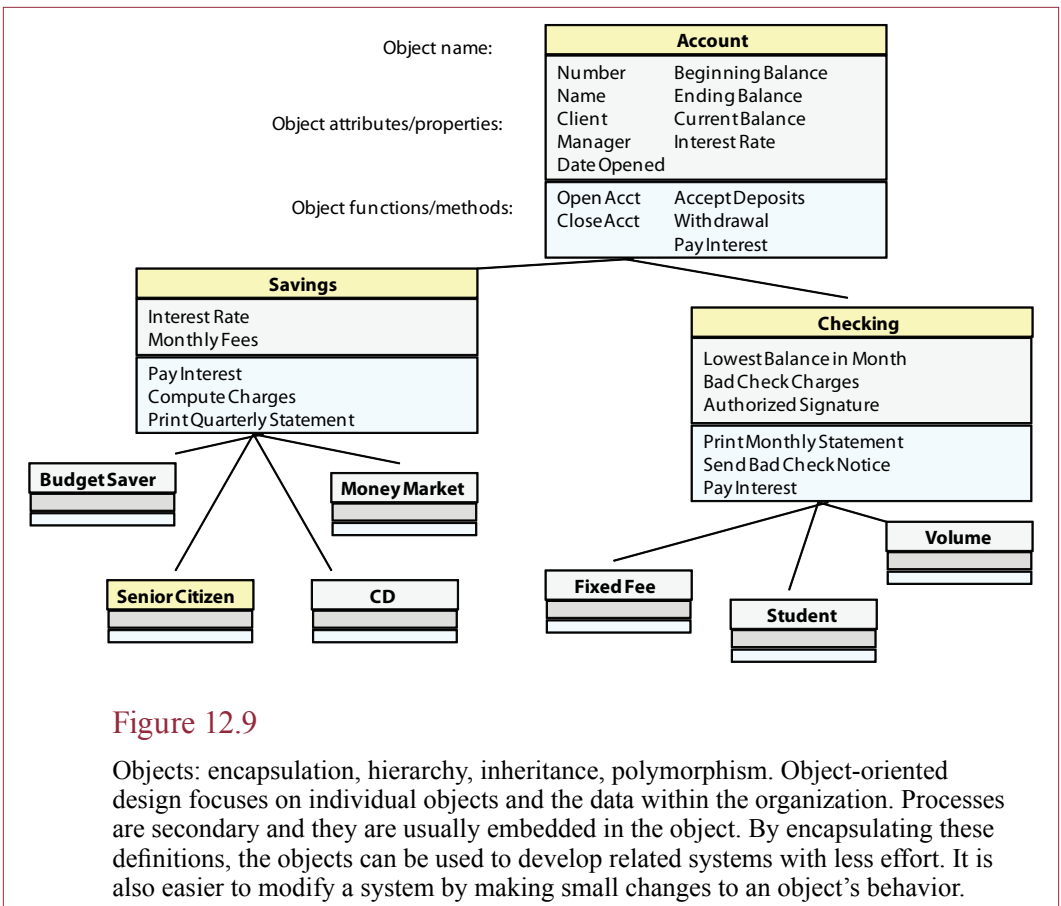


Figure 12.9

Objects: encapsulation, hierarchy, inheritance, polymorphism. Object-oriented design focuses on individual objects and the data within the organization. Processes are secondary and they are usually embedded in the object. By encapsulating these definitions, the objects can be used to develop related systems with less effort. It is also easier to modify a system by making small changes to an object’s behavior.

the small example of a banking system. One primary object will be Accounts. A generic account object would have basic properties such as Account Number, Account Name, Client, Manager, Date Opened, Beginning Balance, Current Balance, and Interest Rate. Each object also has functions, which describe actions that can be performed by the objects and define how to alter the object. In the bank example, there would be functions to Open Account, Close Account, Accept Deposits, Pay Withdrawals, and Pay Interest. Note that each type of account could have a different method for computing interest payments. One account might compound them daily, another weekly, and so on. With the object-oriented approach, the properties and functions are combined into the definition of the object. The goal is to describe a system so that if you change a function, you only have to change one object. All of the other objects and processes remain the same.

Objects are related to each other. Typically there is a base class of objects, and other objects are derived from the base definitions by adding properties and altering functions. This process results in an **object hierarchy**, illustrated in Figure 12.9, that shows how the classes are derived from each other. The bank example has several types of accounts with each of these categories containing further subdivisions.

Figure 12.9 also shows detail in the classes by including some of the properties and member functions. The accounts have elements in common that are an

- * Technical measures
 - * 2-5 times over budget
 - * 2-5 times behind schedule
 - * Missing technical objectives
- * Design problems
 - * Duplication of efforts
 - * Incompatibilities
 - * User/designer conflicts

\$

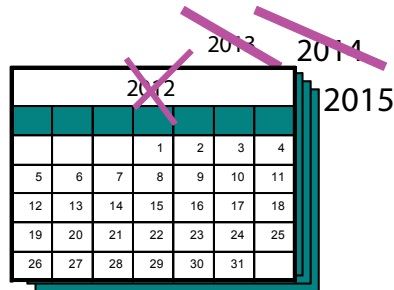
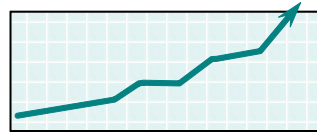


Figure 12.10

Runaway projects. Managers fear runaway projects, but they still occur. Some projects end up two to five times over budget and behind schedule. Some projects are canceled because they never meet their objectives. Some fail because of design problems and conflicts among users, management, and developers. An important step in managing projects is to identify when the project becomes a runaway project.

inheritance from the base class (account), such as the balance attributes. Each level adds additional detail. Each account class also contains member functions to perform operations, such as paying interest. Because the interest computations can be different for each of the accounts, the method is stored with the original definition of each account.

To see the usefulness of the object approach, consider what happens if the bank decides to collect additional data for the checking accounts. The only change needed is to add the new data items (and the associated functions) to the checking account class. All checking accounts will then inherit those properties and functions. None of the other operations are affected. Changes to the information system will only affect the specific accounts; the rest of the system will remain the same.

From Programming to Development

Now that you have an idea of what it takes to create even a simple program, multiply that task by thousands to get a feel for the challenges of building a large, complex application. Plus, the application needs to do what the users want, not necessarily what the programmers want it to do. So someone has to identify business needs and translate those into concepts that the programmers understand. This role is handled by a **systems analyst**. In small projects, the programmers also perform the systems analyst tasks, but in large projects the discussion and design issues are handled by specialists.

In theory, a single programmer could develop almost any application—but you will probably have to wait several years for the project to finish. Keep in mind that the development tools are continually improving, so actually a single programmer today can build a relatively complex system in a short period of time. Projects

that once required teams of programmers weeks or months to build can be built in a short time by even a single programmer. However, there are dozens of other details that need to be handled, including testing, installation, documentation, and training. The overall time can be reduced if additional people are hired to handle these tasks, and if the programming can be divided into pieces so additional programmers can work on the pieces at the same time. Whenever you add people to a project, you need a way to assign tasks, communicate ideas, monitor progress, and control the entire process. At this point, you need a development methodology.

Systems Development Life Cycle

How do you control a major development project? Runaway projects illustrated in Figure 12.10 are a substantial problem in any development effort, but they are particularly important for new designs. Building a project from scratch means it is hard to estimate the amount of time and effort needed to build the system. As projects become larger, they become more difficult to monitor and control. Several major projects, including e-commerce firms, failed because they were unable to produce a working system.

A factor in many runaway projects is the concept of **scope creep** or expanding features. Once development starts, users and programmers start thinking of new ideas that they would like to see in the project. So a simple, two-month project for one division suddenly expands into a two-year companywide project costing millions of dollars. A key role of any IT project manager is to politely avoid adding features that are not immediately necessary.

SDLC was designed to overcome the problems that arose with large projects that involve many users and require thousands of hours of development by multiple analysts and programmers.

Before the use of the SDLC method, several related problems were common. It was hard to coordinate and control the various programmers and analysts, so efforts were duplicated. Individual programmers created portions of a system that would not work together. Users did not always have much input into the process. When they did have input, there were conflicts between users, and analysts did not know which approach to use. With long-term projects, programmers were promoted to other areas or left for different companies. New employees had to learn the system and determine what others had done before they could proceed. Similarly, new users would appear (through promotions and transfers), and existing users would change the specifications of the system. These problems often lead to runaway projects—projects that are significantly late and over budget. Even today, there are many instances of runaway projects.

Figure 12.11 shows some of the main risks in software development as outlined by Boehm in 1991. The one at the bottom of the list is an interesting question because it keeps changing. Developers and academics know how to solve certain types of problems. Computer scientists also know some problems that do not yet have good solutions. Any project that pushes these boundaries is going to be difficult to create and hard to control. Organizations need to be careful when entering into projects that are experimental. Dealing with large projects that use standard tools is hard enough to control. Fortunately, most business projects use relatively common technologies. Still, it is easy to get carried away and start believing that computers can perform impossible tasks.

These problems are related through the issue of control. It is impossible to prevent users from changing the specifications and to prevent employees from tak-

- Personnel shortfalls
- Unrealistic schedules and budgets
- Developing the wrong functions and properties
- Developing the wrong user interface
- Gold plating (adding more functionality/features than necessary)
- Continuing stream of requirements changes (scope creep)
- Shortfalls in externally furnished components
- Shortfalls in externally performed tasks
- Real-time performance shortfalls
- Straining computer-science capabilities

Figure 12.11

Project Risks (Boehm 1991). Some of the main risks that can delay or prevent the successful completion of software development projects. Good development practices can reduce some of these risks, but project managers must constantly watch for signs of problems.

ing other jobs. Likewise, large projects involving many analysts and programmers will always have problems with coordination and compatibility. The goal of SDLC was to design a system that can handle all of these problems.

A key value in SDLC is project management. An important aspect of project management consists of identifying the dependencies among the various tasks. Project management tools exist to help evaluate these dependencies and show how the overall schedule is affected by delays in individual tasks.

Introduction to SDLC

An important feature of the SDLC approach is that it is a comprehensive method. Some organizations (such as EDS) that specialize in systems development have hundreds of pages in manuals to detail all the steps and rules for using SDLC. Fortunately, it is possible to understand SDLC by looking at a smaller number of steps. As illustrated in Figure 12.12, the SDLC approach encompasses five basic stages: (1) feasibility and planning, (2) systems analysis, (3) systems design, (4) implementation, and (5) maintenance and review.

Actually, just about any systems-development methodology uses these five steps. They differ largely in how much time is spent in each section, who does the work, and in the degree of formality involved. The SDLC approach is by far the most formal method, so it offers a good starting point in describing the various methodologies.

Feasibility and Planning

The primary goal of **systems analysis** is to identify problems and determine how they can be solved with a computer system. In formal SDLC methodologies, the first step in systems analysis is a **feasibility study**. A feasibility study is a quick examination of the problems, goals, and expected costs of the system. The objective is to determine whether the problem can reasonably be solved with a computer system. In some cases, maybe there is a better (or cheaper) alternative, or perhaps the problem is simply a short-term annoyance and will gradually disappear. In other cases, the problem may turn out to be more complex than was thought and to involve users across the company. Also, some problems may not be solv-

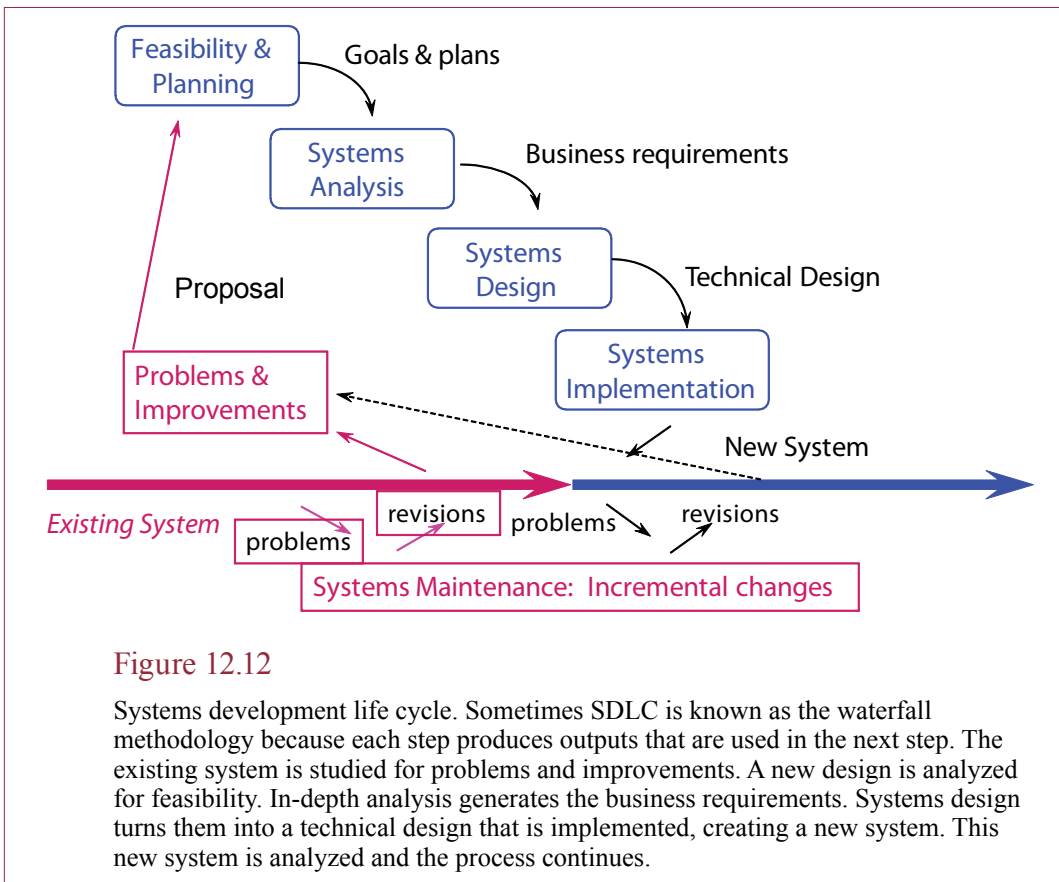


Figure 12.12

Systems development life cycle. Sometimes SDLC is known as the waterfall methodology because each step produces outputs that are used in the next step. The existing system is studied for problems and improvements. A new design is analyzed for feasibility. In-depth analysis generates the business requirements. Systems design turns them into a technical design that is implemented, creating a new system. This new system is analyzed and the process continues.

able with today's technology. It might be better to wait for improved technology or lower prices. In any case, you need to determine the scope of the project to gain a better idea of the costs, benefits, and objectives.

The feasibility study is typically written so that it can be easily understood by nonprogrammers. It is used to “sell” the project to upper management and as a starting point for the next step. Furthermore, it is used as a reference to keep the project on track, and to evaluate the progress of the MIS team. Projects are typically evaluated in three areas of feasibility: economical, operational, and technical. Is the project cost effective or is there a cheaper solution? Will the proposed system improve the operations of the firm, or will complicating factors prevent it from achieving its goals? Does the technology exist, and does the firm have the staff to make the technology work?

When the proposal is determined to be feasible, the MIS team leaders are appointed, and a plan and schedule are created. The schedule contains a detailed listing of what parts of the project will be completed at each time. Of course, it is extremely difficult to estimate the true costs and completion dates. Nonetheless, the schedule is an important tool to evaluate the status of the project and the progress of the MIS teams. As shown in Figure 12.13, planning and scheduling provides the blueprint or structure for the rest of the project. It is a crucial step that provides control for the remaining project.

Detailed work plan
 Performance targets
 Practices & procedures
 User input & control

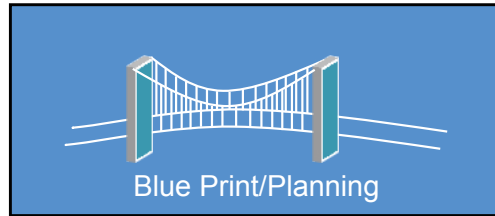


Figure 12.13

Development controls. A complex system requires careful management. Without planning and control, any project will become a runaway. Control begins with a detailed plan and performance targets that enable managers to evaluate progress and identify problems. System control is provided by standardized practices and procedures to ensure that teams are producing compatible output. User input and control ensure that the final project will actually be useful.

Systems Analysis

Once a project has been shown to be feasible and is approved, work can begin on a full-fledged analysis. The first step is to determine how the existing system works and where the problems are located. The technique is to break the system into pieces. Smaller pieces are easier to understand and to explain to others. Also, each piece can be assigned to a different MIS team. As long as they work from the same initial description and follow all of the standards, the resulting pieces should fit back together. Of course, it still takes time and effort to integrate all of the pieces. The key objective in this stage is to understand the business organization and determine the specific requirements for the new project. It is also useful to collect test cases. Modern development systems can be programmed to automatically run test cases as the code is being developed and changed. The test data (with the correct results) ensure that the system always produces accurate results.

Diagrams are often created to illustrate the system. The diagrams are used to communicate among analysts and users, other analysts, and eventually the programmers. Data flow diagrams are a common method to display the relationships that were determined during systems analysis. The diagrams represent a way to divide the system into smaller pieces.

Graphics tools provide a useful way to communicate with the user and to document the user requirements. However, they do not speed up the development process. Producing, changing, and storing documentation can be a significant problem. Yet these tools are necessary because they make it easier for the user

Reality Bytes: What Tech Skills do Employers Want

Because technology continually changes, students are always asking what skills are needed. Actually, even experienced IT employees ask the same question. In 2011, employers said they wanted the several basic skills: 77 percent want programming skills, 82 percent want database skills, 76 percent want analytical and architectural skills, and 80 percent want general problem solving and technical skills. Many employers also want experience, but when the job market tightens, they have few options. In the same study, 50 percent said they were willing to hire new IT graduates; but two-thirds wanted at least some college internship experience.

Adapted from Michael Cooney, "IT Graduates Not 'Well-Trained, Ready-to-go,'" *Network World*, February 25, 2011.

to control the final result. One increasingly common solution is to keep all of the documentation on the computer. This method reduces the costs, makes it easier for everyone to share the documentation, and ensures that all users have up-to-date information for the system.

At the end of the analysis phase, the MIS team will have a complete description of the business requirements. The problems and needs are documented with text, data flow diagrams, and other figures depending on the methodology followed.

Systems Design

The third major step of the SDLC approach is to design the new system. During this step, the new system is typically designed. The objective of systems design is to describe the new system as a collection of modules or subsystems. By subdividing the total project, each portion can be given to a single programmer to develop. As the pieces are completed, the overall design ensures that they will work together. Typically, the diagrams created during the analysis phase can be modified to indicate how the new system will work. The design will list all of the details, including data inputs, system outputs, processing steps, database designs, manual procedures, and feedback and control mechanisms. Backup and recovery plans along with security controls will be spelled out to ensure that the database is protected.

In traditional SDLC methods, managers and users will be shown various components of the system as they are completed. The managers will have to *sign off* on these sections to indicate that they meet the user needs. This signature is designed to ensure that users provide input to the system. If there are many diverse users, there can be major disagreements about how the system should function. Sign-offs require users to negotiate and formally agree to the design. It is relatively easy to make design changes at this stage. If everyone attempts to make changes at later stages, the cost increases dramatically.

In terms of physical design, some of the hardware and software will be purchased. Programmers will write and test the program code. In most large projects, the actual coding takes only 15 to 30 percent of the total development time. Initial data will be collected or transferred from existing systems. Manuals and procedures will be written to instruct users and system operators on how to use the system.

Reality Bytes: You Won. Just Kidding

In 1994, the U.S. Congress created an international green-card lottery. The annual competition randomly selects 50,000 people to live and work legally in the U.S. with minimal other criteria. Almost 15 million people applied for the lottery in 2011. In May, the State Department posted a list of 20,000 winning numbers on its Web site. A few lucky people, such as Max, a 28-year-old German were ecstatic. Until a few days later. The State Department identified a glitch in the computer program that selected the names. Apparently, 90 percent of the numbers selected had applied in the first two days. Deputy assistant secretary of state David Donahue noted that “These results are not valid because they did not represent a fair, random selection of the entrants as required by U.S. law.” The lottery has been conducted electronically for 15 years, but the State department blamed a coding error in a new computer program. The department did not provide an explanation as to why no one tested the program ahead of time. The lottery was rescheduled.

Adapted from *The Wall Street Journal*, “Computer Glitch Voids Green-Card Lottery,” May 14, 2011.

Design tools can be used to create prototypes of major system elements. For example, a designer can quickly piece together displays that illustrate how each screen might look and how the user will see the system. These prototypes can be used to help users walk through aspects of the proposed system and make changes while it is easy and inexpensive. The walkthroughs also provide management with feedback regarding the time schedule and anticipated costs of the project, because they are often scheduled in the original feasibility study.

The output of the design stage consists of a complete technical specification of the new system. It includes as many details as possible, sometimes leading to thousands of pages (or computer files) of description.

One of the difficulties in the design stage is sometimes called *creeping elegance*. As the system is being built, analysts, programmers, and users all want to include additional features. Although many of the features are good ideas, the continual evolution of the system causes additional delays. It also complicates testing, because changes in one section can affect the rest of the system.

Systems Implementation

Systems implementation involves installation and changeover from the previous system to the new one, including training users and making adjustments to the system. Many nasty problems can arise at this stage. You have to be extremely careful in implementing new systems. First, users are probably nervous about the change already. If something goes wrong, they may never trust the new system. Second, if major errors occur, you could lose important business data.

A crucial stage in implementation is final testing. Testing and quality control must be performed at every stage of development, but a final systems test is needed before staff entrust the company’s data to the new system. Occasionally, small problems will be noted, but their resolution will be left for later. In any large system, errors and changes will occur. The key is to identify them and determine which ones must be fixed immediately. Smaller problems are often left to the software maintenance staff.

Technology Toolbox: Creating Forms with InfoPath

Problem: You want employees to use digital forms to collect basic data such as expense reports.

Tools: You can use InfoPath in Microsoft Office to create, exchange, and store data from digital forms.

InfoPath is included with several business versions of Office 2007. It has a forms-builder tool to create forms and define the data that will be collected. Once the form is created, it can be published to a SharePoint server, e-mailed to people, or built into a Visual Studio project. Simple forms can be built that can be opened with just a Web browser, so recipients do not need InfoPath installed on their computers. However, InfoPath forms are more powerful than browser forms.

Creating forms is relatively easy—especially if you can use one of the sample forms included with InfoPath. The basic steps are to define the data elements, then add text and input boxes using the layout tools. After checking the design for errors, you publish the form so employees can fill it out as needed.

The easiest method is to centralize the forms and store them on a SharePoint server. You can use additional tools to build in workflow procedures. The expense report is one of several sample forms included with InfoPath. Employees go to the SharePoint server and create a new form filling out the relevant data. The data is saved on the SharePoint server, and the form is e-mailed automatically to the employee's manager for review. At any time, the manager can see summaries of specified data columns on the SharePoint server site. All data is stored in XML files, so it can be transferred to other systems fairly easily.

 Subtotal: 1,046.64 | Less cash advance: 0.00 | Total expenses: 1,046.64 |

Quick Quiz:

1. What standard business forms would you want to create electronically?
2. What security conditions would you impose when installing expense report forms on a SharePoint server?
3. What are the benefits of using digital forms instead of paper forms?

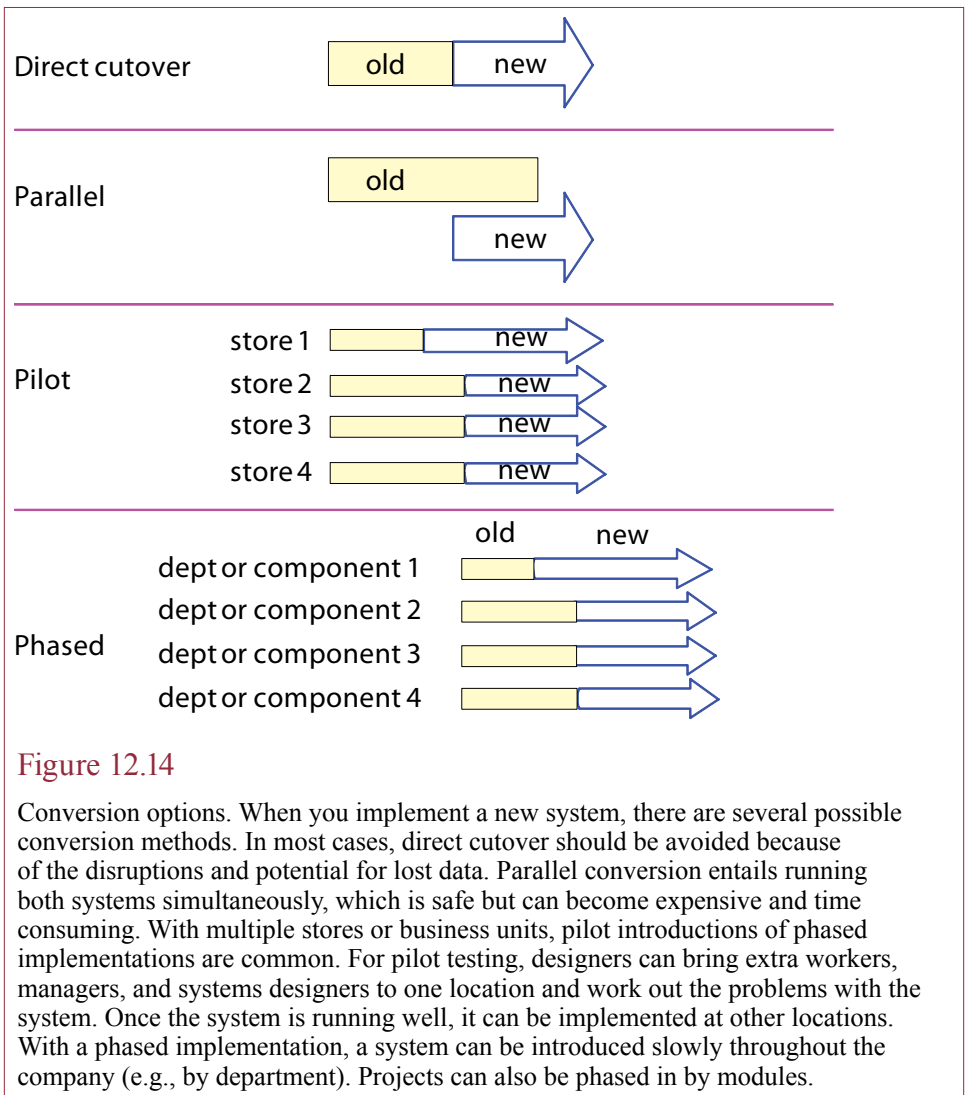


Figure 12.14

Conversion options. When you implement a new system, there are several possible conversion methods. In most cases, direct cutover should be avoided because of the disruptions and potential for lost data. Parallel conversion entails running both systems simultaneously, which is safe but can become expensive and time consuming. With multiple stores or business units, pilot introductions of phased implementations are common. For pilot testing, designers can bring extra workers, managers, and systems designers to one location and work out the problems with the system. Once the system is running well, it can be implemented at other locations. With a phased implementation, a system can be introduced slowly throughout the company (e.g., by department). Projects can also be phased in by modules.

Change is an important part of MIS. Designing and implementing new systems often cause changes in the business operations. Yet many people do not like changes. Changes require learning new methods, forging new relationships with people and managers, or perhaps even loss of jobs. Changes exist on many levels: in society, in business, and in information systems. Changes can occur because of shifts in the environment, or they can be introduced by internal **change agents**. Left to themselves, most organizations will resist even small changes. Change agents are objects or people who cause or facilitate changes. Sometimes it might be a new employee who brings fresh ideas; other times changes can be mandated by top-level management. Sometimes an outside event such as arrival of a new competitor or a natural disaster forces an organization to change. Whatever the cause, people tend to resist change. However, if organizations do not change, they cannot survive. The goal is to implement systems in a manner that recognizes resistance to change but encourages people to accept the new system. Effective

implementation involves finding ways to reduce this resistance. Sometimes, implementation involves the cooperation of outsiders such as suppliers.

Because implementation is so important, several techniques have been developed to help implement new systems. Direct cutover is an obvious technique, where the old system is simply dropped and the new one started. If at all possible, it is best to avoid this technique, because it is the most dangerous to data. If anything goes wrong with the new system, you run the risk of losing valuable information because the old system is not available. The various methods are displayed in Figure 12.14.

In many ways, the safest choice is to use parallel implementation. In this case, the new system is introduced alongside the old one. Both systems are operated at the same time until you determine that the new system is acceptable. The main drawback to this method is that it can be expensive because data has to be entered twice. In addition, if users are nervous about the new system, they might avoid the change and stick with the old method. In this case, the new system may never get a fair trial.

Several intermediate possibilities are called **phased implementation**. For example, if you design a system for a chain of retail stores, you could pilot-test the first implementation in one store. By working with one store at a time, there are likely to be fewer problems. But if problems do arise, you will have more staff members around to overcome the obstacles. When the system is working well in one store, you can move to the next location. Similarly, even if there is only one store, you might be able to split the implementation into sections based on the area of business. You might install a set of computer cash registers first. When they work correctly, you can connect them to a central computer and produce daily reports. Next, you can move on to annual summaries and payroll. Eventually the entire system will be installed.

Maintenance

Once the system is installed, the MIS job has just begun. Computer systems are constantly changing. Hardware upgrades occur continually, and commercial software tools may change every year. Users change jobs. Errors may exist in the system. The business changes, and management and users demand new information and expansions. All of these actions mean the system needs to be modified. The job of overseeing and making these modifications is called **software maintenance**.

The pressures for change are so great that in most organizations today as much as 80 percent of the MIS staff is devoted to modifying existing programs. These changes can be time consuming and difficult. Most major systems were created by teams of programmers and analysts over a long period. In order to make a change to a program, the programmer has to understand how the current program works. Because the program was written by many different people with varying styles, it can be hard to understand. Finally, when a programmer makes a minor change in one location, it can affect another area of the program, which can cause additional errors or necessitate more changes.

One difficulty with software maintenance is that every time part of an application is modified, there is a risk of adding defects (bugs). Also, over time the application becomes less structured and more complex, making it harder to understand. At some point, a company may decide to replace or improve the heavily modified system. Several techniques can be used to improve an existing system, ranging

Feasibility Comparison	
Cost & Budget	Compare actual costs to budget estimates.
Time Estimates	Was project completed on time?
Revenue Effects	Does system produce additional revenue?
Maintenance Costs	How much money and time are spent on changes?
Project Goals	Does system meet the initial goals of the project?
User Satisfaction	How do users (and management) evaluate the system?
System Performance	
System Reliability	Are the results accurate and on time?
System Availability	Is the system available on a continuous basis?
System Security	Does the system provide access only to authorized users?

Figure 12.15

Evaluation of completed projects. When projects are completed, the design team should evaluate the project and assess the development procedures. Cost and time estimates can be used to improve estimates for future projects. System performance issues can be addressed with future upgrades. It is important that the system achieve project goals and provide users with necessary tools and support.

from rewriting individual sections to restructuring the entire application. The difference lies in scope—how much of the application needs to be modified. Older applications that were subject to modifications over several years tend to contain code that is no longer used, poorly documented changes, and inconsistent naming conventions. These applications are prime candidates for restructuring, during which the entire code is analyzed and reorganized to make it more efficient. More important, the code is organized, standardized, and documented to make it easier to make changes in the future.

Evaluation

An important phase in any project is evaluating the resulting system. As part of this evaluation, it is also important to assess the effectiveness of the particular development process. There are several questions to ask: Were the initial cost estimates accurate? Was the project completed on time? Did users have sufficient input? Are maintenance costs higher than expected? The assessment items are summarized in Figure 12.15.

Evaluation is a difficult issue. As a manager, how can you tell the difference between a good system and a poor one? In some way, the system should decrease costs, increase revenue, or provide a competitive advantage. Although these effects are important, they are often subtle and difficult to measure. The system should also be easy to use and flexible enough to adapt to changes in the business. If employees or customers continue to complain about a system, it should be reexamined.

A system also needs to be *reliable*. It should be available when needed and should produce accurate output. Error detection can be provided in the system to recognize and avoid common problems. Similarly, some systems can be built to tolerate errors, so that when errors arise, the system recognizes the problem

Strengths	Weaknesses
Control.	Increased development time.
Monitor large projects.	Increased development costs.
Detailed steps.	Systems must be defined up front.
Evaluate costs and completion targets.	Rigidity.
Documentation.	Hard to estimate costs, project overruns.
Well-defined user input.	User input is sometimes limited.
Ease of maintenance.	
Development and design standards.	
Tolerates changes in MIS staffing.	

Figure 12.16

Strengths and weaknesses of SDLC. The SDLC methodologies were created to control large, complex development projects. They work fairly well for those types of processes. They do not work as well for small projects that require rapid development or heavy user involvement with many changes.

and works around it. For example, some computers exist today that automatically switch to backup components when one section fails, thereby exhibiting **fault tolerance**.

An important concept for managers to remember when dealing with new systems is that the evaluation mechanism should be determined at the start of the project. Far too often, the question of evaluation is ignored until someone questions the value of the finished product. It is a good design practice to ask what would make this system a good system when it is finished, or how we can tell a good system from a bad one in this application. Even though these questions may be difficult to answer, they need to be asked. The answers, however incomplete, will provide valuable guidance during the design stage.

Recall that every system needs a goal, a way of measuring progress toward that goal, and a feedback mechanism. Traditionally, control of systems has been the task of the computer programming staff. Their primary goal was to create error-free code, and they used various testing techniques to find and correct errors in the code. Today, creating error-free code is not a sufficient goal.

Everyone has heard the phrase “The customer is always right.” The meaning behind this phrase is that sometimes people have different opinions on whether a system is behaving correctly. When there is a conflict, the opinion that is most important is that of the customer. In the final analysis, customers are in control because they can always take their business elsewhere. With information systems, the users are the customers and the users should be the ones in control. Users determine whether a system is good. If the users are not convinced that the system performs useful tasks, it is not a good system.

Strengths and Weaknesses of SDLC

The primary purpose of the SDLC method of designing systems is to provide guidance and control over the development process. As summarized in Figure 12.16, there are strengths and weaknesses to this methodology. SDLC management control is vital for large projects to ensure that the individual teams work

together. There are also financial controls to keep track of the project expenses. The SDLC steps are often spelled out in great detail. The formality makes it easier to train employees and to evaluate the progress of the development. It also ensures that steps are not skipped, such as user approval, documentation, and testing. For large, complex projects, this degree of control is necessary to ensure the project can be completed. Another advantage of SDLC is that by adhering to standards while building the system, programmers will find the system easier to modify and maintain later. The internal consistency and documentation make it easier to modify. With 80 percent of MIS resources spent on maintenance, this advantage can be critical.

In some cases the formality of the SDLC approach causes problems. Most important, it increases the cost of development and lengthens the development time. In many cases less than 25 percent of the time is spent on actually writing programs. A great deal of the rest of the time is spent filling out forms and drawing diagrams.

The formality of the SDLC method also causes problems with projects that are hard to define. SDLC works best if the entire system can be accurately specified in the beginning. That is, users and managers need to know exactly what the system should do long before the system is created. That is not a serious problem with transaction-processing systems. However, consider the development of a complex decision support system. Initially, the users may not know how the system can help. Only through working with the system on actual problems will they spot errors and identify enhancements.

Although some large projects could never have been completed without SDLC, its rigidity tends to make it difficult to develop many modern applications. Moreover, experience has shown that it has not really solved the problems of projects being over budget and late. As a result of this criticism, many people are searching for alternatives. One possibility is to keep the basic SDLC in place and use technology to make it more efficient. Other suggestions have been to replace the entire process with a more efficient development process, such as prototyping. Consider the assistance of technology first.

Several researchers at Carnegie Mellon University have created (and trademarked) the **capability maturity model integration (CMMI)** to help development organizations evaluate their abilities. Figure 12.17 shows the various levels of maturity. The goal is to improve the development process within an organization so that everyone follows a process that is measurable and sustainable. In standard management terms, quantifying the development process makes it possible to fine-tune and improve. Possibly the greatest strength of the CMMI approach is also one of its weaknesses. The overall approach is designed to support and encourage mediocrity in development. Programmers are considered interchangeable—an organization that succeeds by relying on “star” programmers is considered to be inferior. For some large organizations (particularly governments), this characterization makes sense—the system should function even with staff turnover. The weakness is that some software development requires creativity and flexibility to create new approaches.

Alternatives to SDLC

Is SDLC always the best approach? What other methodologies could be used? The two primary drawbacks to SDLC are that (1) it takes a considerable amount of time, and (2) all the system details have to be specified

1. **Initial.** Ad hoc development with undefined processes. Often driven by individual programmers.
2. **Managed.** Standard project management tools to track costs and schedules. Basic processes to ensure development is repeatable.
3. **Defined.** Management and development is defined and standardized. Processes are documented and followed.
4. **Quantitatively Managed.** Detailed measures are collected and evaluated.
5. **Optimizing.** Continuous improvement methods are applied to fine-tune and improve the development process.

Figure 12.17

Capability Maturity Model. Based on standard management techniques. A development organization should strive to install processes, measure progress, and improve the development methodology.

up front. The project management and control features add paperwork and delays, making SDLC unsuitable for small projects. SDLC works reasonably well for transaction-processing systems that are well defined where the design elements can be specified up front. It does not work well for decision support systems particularly when users do not really know exactly what they want the system to do.

Prototyping or Iterative Development

Prototyping has been proposed as a method to use for systems that are not overly complex and do not involve too many users or analysts. Just as automobile engineers design prototypes before attempting to build the final car, MIS programmers can build early versions of systems. These systems are then continually modified until the user is satisfied.

The first step in designing a system via prototyping is to talk with the user. The analyst then uses a fourth-generation language and a DBMS to create approximately what the user wants. This first step generally requires only a couple of weeks. The business user then works with the prototype and suggests changes. The analyst makes the changes and this cycle repeats until the user is satisfied or decides that the system is not worth pursuing. The emphasis is on getting a working version of the system to the user as fast as possible, even if it does not have all the details. Figure 12.18 illustrates the cycle involved in prototyping.

The major advantage of prototyping is that users receive a working system much sooner than they would with the SDLC method. Furthermore, the users have more input so they are more likely to get what they wanted. Finally, remember that a large portion of MIS time is spent making changes. A system designed with the prototyping method is much easier to change because it was designed to be modified from the start.

Extreme Programming and Agile Development

In some ways, **extreme programming (XP)** is a new concept; in other ways it is an extension of the prototyping ideas. The main premise of XP is that SDLC and its variants are too large and cumbersome. While they might provide control, they end up adding complexity, taking more time, and slowing down top programmers.

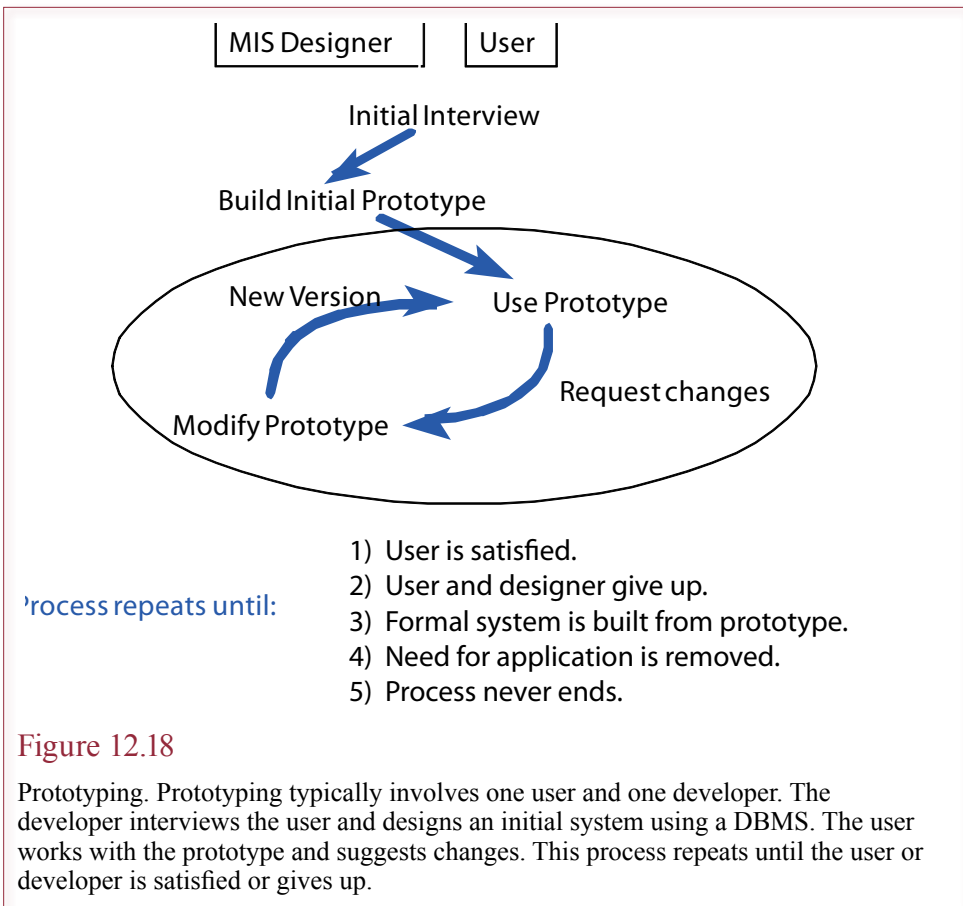


Figure 12.18

Prototyping. Prototyping typically involves one user and one developer. The developer interviews the user and designs an initial system using a DBMS. The user works with the prototype and suggests changes. This process repeats until the user or developer is satisfied or gives up.

XP and agile development simplify the development process by focusing on small releases (similar to prototyping) that provide value to the customer.

As shown in Figure 12.19, XP and agile development were pushed heavily in the development of new Web-based systems. In this highly-competitive environment, getting applications out the door and on the Web quickly was more important than loading on tons of features. Yet, everything had to work correctly. To improve quality XP adopted a relatively new principle from computer science: building test cases first. The basic system is designed in terms of what each module should accomplish. These features are defined with a set of test cases. Programmers then write code and feed the test cases through to ensure the modules work correctly. Whenever the system is changed, the programmers rerun the test cases to ensure nothing else was broken. Tools were developed to store the test cases and the results, making it easy to develop new cases and modules quickly.

One new aspect to XP is paired programming, where two programmers work together constantly. Generally, one is the lead programmer and the other is responsible for testing, but the jobs can overlap and be defined by the team. Making testing a key element of programming is an important part of XP. However, paired programming is seen by many as an inefficient use of resources. The second programmer is often a less experienced developer and can slow down an experienced developer. Besides, it can be more efficient to have one person test large sections of code at a time, instead of multiple people testing separate pieces.

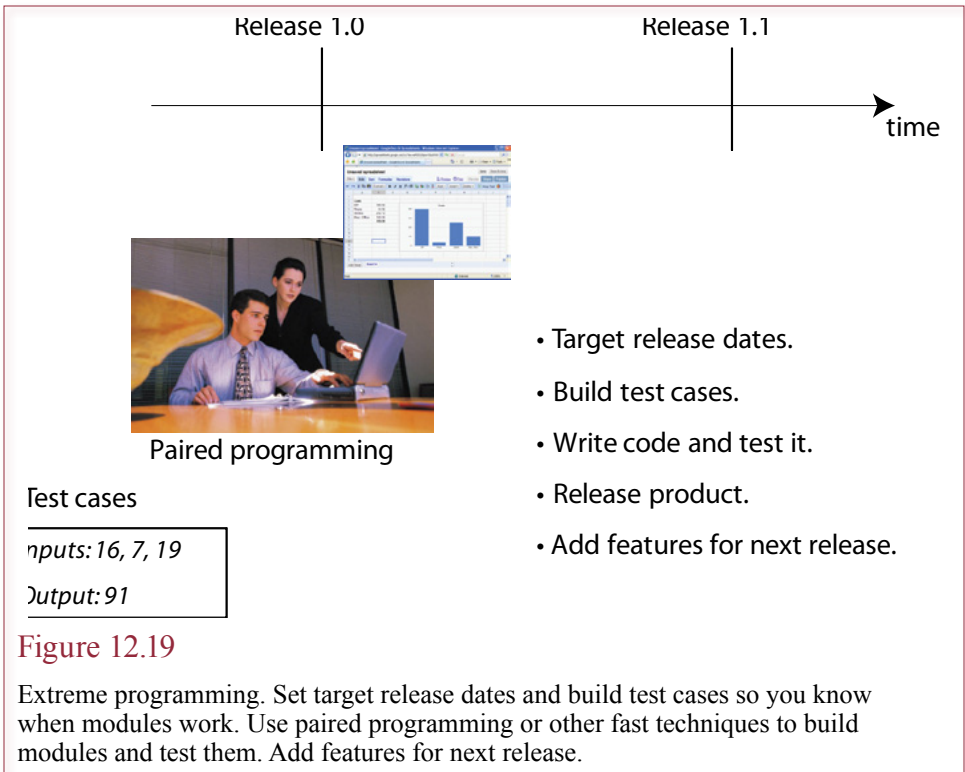


Figure 12.19

Extreme programming. Set target release dates and build test cases so you know when modules work. Use paired programming or other fast techniques to build modules and test them. Add features for next release.

One of the most challenging aspects to development is that there is a tremendous difference between individual programmers—in subject area knowledge, speed of programming, number of defects, and code maintainability. Some methodologies work well when an organization has top-notch developers, but fall apart in other companies. In choosing a methodology, managers must be aware of the capabilities of the individual programmers—and beware of turnover.

Developing Systems Requires Teamwork: JAD and RAD

Designing and developing systems is much easier if the entire system can be built by one person. In fact, that is one of the strengths of recent tools—they enable a single person to build more complex systems. However, many information systems, especially those that affect the entire organization, require teams of IS workers. As soon as multiple designers, analysts, and programmers are involved, everyone encounters management and communication problems. MIS researchers have measured the effects of these problems. One study by DeMarco and Lister showed that on large projects, 70 percent of a developer's time is spent working with others. Jones noted that team activities accounted for 85 percent of the development costs. There seem to be substantial areas for improvement in systems development by focusing on teamwork.

One of the most difficult steps in creating any new system is determining the user requirements. What does the system need to do and how will it work? This step is crucial. If the designers make a mistake here, the system will either be useless or will need expensive modifications later. Prototyping and SDLC take different approaches to this problem. With SDLC, analysts talk with users and write reports that describe how the system will operate. Users examine the reports and

Technology Toolbox: Programming a New Function in Excel

Problem: You need to add a tricky function to a spreadsheet.

Tools: Microsoft Office contains a language to create your own tools and functions.

```
Function BlackScholes(CallPut As String, StockPrice As Double, _
    ExercisePrice As Double, TimeLeft As Double, rate As Double, _
    volatility As Double) As Double
    Dim d1 As Double, d2 As Double
    d1 = (Math.Log(StockPrice / ExercisePrice) + _
        (rate + volatility ^ 2 / 2) * TimeLeft) / _
        (volatility * Math.Sqrt(TimeLeft))
    d2 = d1 - volatility * Math.Sqrt(TimeLeft)
    If (Left(CallPut, 1) = "c") Then
        BlackScholes = StockPrice _
            * Application.WorksheetFunction.NormSDist(d1) _
            - ExercisePrice * Exp(-rate * TimeLeft) _
            * Application.WorksheetFunction.NormSDist(d2)
    Else
        BlackScholes = ExercisePrice * Exp(-rate * TimeLeft) * _
            Application.WorksheetFunction.NormSDist(-d2) - StockPrice _
            * Application.WorksheetFunction.NormSDist(-d1)
    End If
End Function
```

Many programming languages exist for different jobs, but they all have similar features. The common tasks are (1) define variables and perform calculations, (2) create functions and subroutines, (3) use conditional (if) statements to make choices, (4) write loops to repeat steps, and (5) input and output data (to the spreadsheet in this case). You use these building blocks to build functions and applications.

A financial option is a contract that enables you to purchase (or sell) shares of stock in the future for a specified exercise price. The stock currently trades at some other price, so you are gambling the price will increase or decrease. The Black-Scholes equation from finance is often used to calculate a value for option prices.

You need a place to write your new function. Open a new worksheet. Choose Developer/Record Macro. Start recording, click a cell in the worksheet and stop recording. Now choose View Code and delete the Macro1 subroutine you created in Module1. Add the new BlackScholes function.

Return to the spreadsheet and enter some sample data for stock price (60), exercise price (65), time left (0.25), rate (0.08), and volatility (0.3). To determine the value of a call option, enter the formula: =BlackSholes("call", B2, B3, B4, B5, B6), where the cell values match the locations you put the sample data. The spreadsheet will call your new function and return the results.

stock price	60	call	2.133368
exercise price	65	put	5.846282
time left	0.25		
rate	0.08		
volatility	0.3		

Quick Quiz:

1. What does the statement `If (Left(CallPut, 1) = "c")` do in the code?
2. What security setting do you need for this function to work?
3. How can a function directly alter several cells in a spreadsheet?

Get everyone together to identify the primary elements of the design with no distractions.



Figure 12.20

Joint application design. Application design can be accelerated and simplified by putting key users and developers together for a few days. By focusing on the single project, everyone gets input and can reach a consensus in a shorter time.

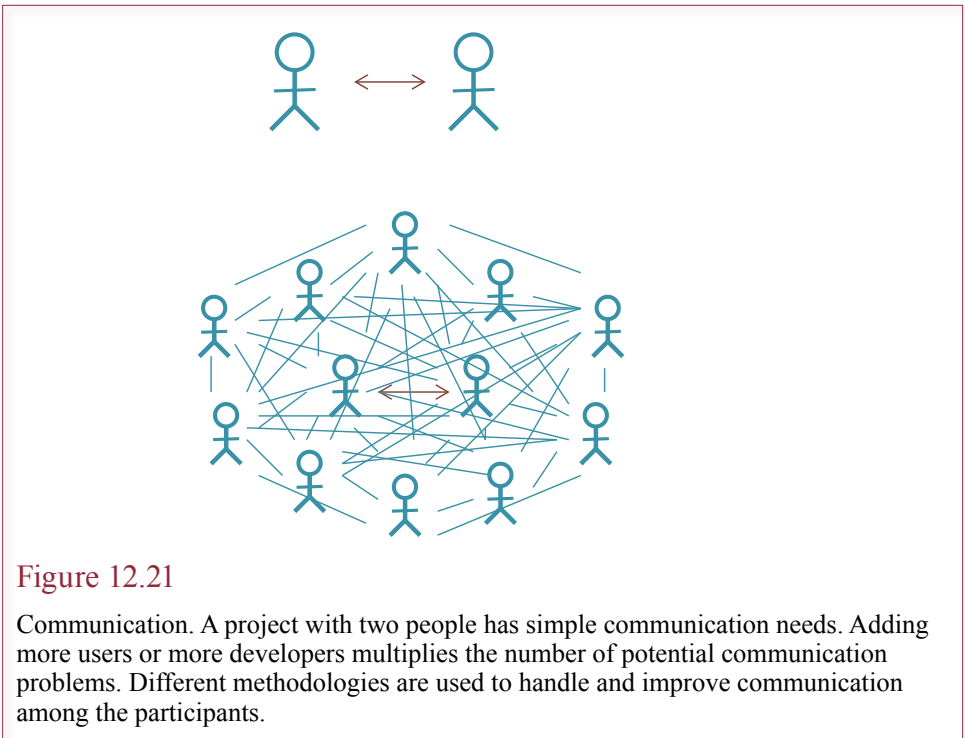
make changes. This approach is time consuming and difficult for users because they only see paper notes of the proposed system. Prototyping overcomes some of the problems by letting users work with actual screens and reports. But use of prototyping is hard to expand beyond one or two users.

Some companies overcome the problems of SDLC by prototyping each input screen and report with one or two primary users. Once the main concepts have been designed, the analysts formalize the system and get approval from other users. The designs are then given to programmers to create with the traditional SDLC development methods.

Recall that an important reason for using SDLC is to obtain the views and agreement of many users. Using traditional interview methods and paper documentation, this process often takes several months. Each change has to be reexamined by other users, and disagreements have to be resolved.

A technique known as **joint application design (JAD)** was created to speed up the design stage. With JAD the main system is designed in an intense three-to five-day workshop. As shown in Figure 12.20, users, managers, and systems analysts participate in a series of intense meetings to design the inputs (data and screens) and outputs (reports) needed by the new system.

By putting all of the decision makers in one room at the same time, conflicts are identified and resolved faster. Users and managers gain a better understanding of the problems and limitations of technology. The resulting system has greater



value for users and managers because it more closely matches their needs. There is less need for changes later, when they become more expensive, so the system is cheaper to create.

The biggest drawback to JAD is that it requires getting everyone together at the same time for an extended period of time. Even for moderately complex systems, the meetings can run eight hours a day for three to five days. Most managers (and users) find it difficult to be away from their jobs for that length of time. Higher-level managers are also needed at these meetings to ensure the system provides the appropriate reports and information. Finally, the meetings can succeed only if they are led by a trained facilitator. The facilitator keeps the discussions moving in the right direction, minimizes conflicts, and encourages everyone to participate. At the end of the sessions, the systems development team should have a complete description of the proposed system.

Rapid application development (RAD) applies the value of teamwork to the developers. By providing advanced development tools, prebuilt objects, and collaboration tools, some companies have found it is possible to reduce the overall development time. The key is to target steps that can overlap and be performed by multiple teams. By improving the collaboration tools, more steps can be compressed. Many e-commerce projects were developed with RAD techniques. Firms were concerned about being the first in the market and felt they needed to develop software rapidly. The goal of being first was later shown to be pointless, but the techniques of using small groups of programmers with advanced tools, collaboration, and intense programming sessions were relatively successful at quickly producing thousands of new applications.

Reality Bytes: Thousands of Apps not Much Money

In 2011, Google claimed that 450,000 developers had produced 200,000 applications for Android-based smart phones. In March, 2011, third-party counts estimated 250,000 Android apps versus 350,000 for Apple. But, the count quickly becomes meaningless—no one even has time to look at even a review of that many applications. Distimo, an analytics firm, noted that only 72,000 of the Android apps were sold for money; and of that list, only two had been downloaded more than 500,000 times worldwide. The company reported that of the iPhone's 211,000 paid apps, six were downloaded more than 500,000 times in April and May 2011. In terms of games, the most popular category, the Android market has five paid games downloaded more than 250,000 times and the iPhone has 10 paid games downloaded in the U.S. over two months. Of course, the iPhone has been sold for several more years than the Android phones. For developers and entrepreneurs, the point is that despite the potentially large market, very few smart phone applications make any money. In the Android market, 20 percent of all free apps and 80 percent of all paid apps were downloaded less than 100 times.

Adapted from Clint Boulton, "Android No Threat to Apple in Paid Apps," *eWeek*, May 29, 2011; and <http://www.businessinsider.com/charts-of-the-week-ipad-competition-is-toast-2011-3#google-is-closing-the-gap-on-apples-app-store-3>

Communication

Communication is a challenge in any project—but it is particularly critical in programming or systems development projects. Figure 12.21 shows the basic issue that as the number of participants increases, the need for communication multiplies. One of the key goals of a methodology is to define and improve the way that everyone communicates. Some formal systems, such as SDLC, define specific communication paths—often with a top-down approach where project managers produce daily or weekly statements. The figure also explains why adding people to a project slows it down. More people means more communication needs to take place.

Other methodologies work to reduce the communication needs by reducing the number of people—or by focusing the immediate communication needs among a smaller number of participants. For example, prototyping works because one developer works with one user, emphasizing the input and communication of that person. Extreme programming and modern tools work because they enable a single developer or small team to produce systems instead of relying on a large number of programmers. Even SDLC relies on reducing communication needs by emphasizing the importance of splitting a problem into smaller, independent pieces. Each piece can be developed by one person or a small team—without needing to see all of the details from every other programmer. The goal is to isolate the details and communicate only the formal interface connections.

Object-Oriented Design

Based on the value of OOP, it was natural for organizations to try and extend the concepts of objects to the entire design process. The goal is to define business objects that will apply to the entire organization. The objects would include the ability to retrieve data and perform standard functions. For instance, a Customer

Reality Bytes: Random Hacks of Kindness

Computer programmers can be expensive. Developing software requires training and creativity and demand for programmers is high, so it can be difficult for not-for-profit firms to hire or pay programmers. In 2009, Google, Yahoo!, the World Bank, NASA, and Microsoft created the “Random Hacks of Kindness” organization to encourage programmers and hackers to devote a weekend to helping find solutions to old problems and help not-for-profit organizations. In 2011 in Nairobi, developers stayed on site for 36 hours to develop programs to assist the Kenyan Red Cross manage volunteers. Another application helped people infected with HIV/AIDS stay on their medication during a disaster. An application created in Atlanta was similar to one built in Toronto. MessageCarrier can be used in a disaster to collect messages from people in remote areas without Internet or phone access. When the message phone is reconnected to the network, it will transfer all of the messages.

Adapted from Alden Mahler Levine, “‘Brains Collide’ During Hackathon for Climate Change, Disaster Relief,” *CNN Online*, June 8, 2011.

object could be defined one time and stored in a central location. Whenever a programmer builds an application, the appropriate objects could be retrieved from the central store and pieced together to create the final application. With most of the work defined ahead of time, it would be easier and faster to build the application compared to starting from scratch each time.

Although the idea was sound, object-oriented design did not work very well in practice. Part of the problem is the necessity of defining all of the objects ahead of time. Most organizations are so busy trying to get basic tasks completed they do not have time to create objects that might be used someday. Another problem is that organizations change relatively quickly, often resulting in the need for new and modified objects. Finally, most big organizations simply decided to purchase ERP systems to integrate all of the organizations applications. The ERP systems use a DBMS to store data, and have internally-defined objects that can be used for custom programming. In a sense, the ERP systems solved the problem by pre-defining standard business objects that can be customized for any organization. But the initial development costs are borne by the ERP vendors and spread across thousands of customers. For any other application, OO design has generally been limited to single applications.

Open Source Development

Open source development is an interesting new method of developing complex software. With this approach, developers from virtually any company or location work on portions of the code. Usually, one person coordinates the efforts and identifies major changes and structure. The individual programmers write, debug, or test sections of code. If a programmer finds a better way to implement a function, the newer version is incorporated into the code. Hundreds or even thousands of programmers can contribute to the development of a project.

So far, this methodology has been used only to develop “free” software that is available for everyone’s use. Many of the techniques were pioneered by Richard Stallman who developed emacs, a programmer’s text editor. He later founded the GNU project (www.gnu.org) that uses the same methods to create and distribute

tools and systems software. Linus Torvalds uses a similar approach to create and distribute the Linux operating system.

Open source development is interesting in terms of both the sophisticated software that has been created and the development methodology. Using Internet communications, and only a small team to coordinate and review the work, thousands of individuals have been able to work together to create complex software that rivals commercial products costing millions of dollars to create. In theory, similar techniques could be used to improve development within business. On the other hand, the technique requires the cooperation of hundreds of developers, often some of the best programmers in the world. It might be possible to hire these programmers on a freelance basis. A few companies offer Web sites that enable you to auction contracts for various portions of a programming job. But it is not entirely clear that this approach is cheaper than just hiring the best programmers.

Another issue with open source development also affects your decision about whether to use open source products such as Linux. How is the software going to be maintained and updated? Creating the initial software is only the first step. Bugs have to be fixed and new features added on a regular basis. As long as there is a core group of people willing to continue working on the project, these issues can be handled. Or if you have a staff with the skills to modify the software, you can make any changes you want—because you have the source code. But what happens 20 years later? Or even in the short run, can open source projects devote the time and money to usability testing and radical improvements as hardware changes? A commercial company has a financial incentive and the cash flow to keep products moving forward. Open source development has only the personal motivations of the prime organizer and the world developer community. Sometimes these motivations are enough to ensure the longevity of a product; sometimes they are not.

End-User Development

The term end user development simply means that users do all of the development work themselves. In many ways, it resembles prototyping, except that users (instead of analysts from the MIS department) create and modify the prototypes. Clearly the main advantage is that users get what they want without waiting for an MIS team to finish its other work and without the difficulty of trying to describe the business problems to someone else.

Two basic reasons explain why end-user development is increasingly popular. First, most MIS organizations are facing a two- or three-year backlog of projects. This means that if you bring a new project to MIS, the designers will not even start on it for at least two years (unless you give up some other project). The second reason is that software tools are getting more powerful and easier to use at the same time. Today it is possible for users to create systems with a spreadsheet in a few hours that 10 years ago would have taken MIS programmers a month to build with third-generation languages. As tools become more powerful and more integrated, it becomes possible to create even more complex systems. Think about some of the database and Web tools—such as PivotTables. Ten years ago, most users would not dream of being able to create these reports. Today, you can get reports and analyze data any way you want with a few clicks of the mouse. The advantages of end-user development are similar to those in prototyping. In particular, users get what they want, and they get working systems sooner.

The potential problems of end-user development are not always easy to see. Most of them arise from the fact that users generally lack the training and experience of MIS analysts and programmers. For instance, systems produced by end users tend to be written for only one person to use. They are oriented to working on stand-alone personal computers. The systems are often customized to fit the needs of the original users. Moreover, most users do not write documentation, so others will have difficulty using the products. Because of lack of training, users rarely perform as much testing as they should. The systems lack security controls and are hard to modify. Think about the problems you encounter when you are given a spreadsheet that was created by the person who held the job before you.

Other problems stem from the bottom-up approach inherent in end-user development. People in different areas of the company will wind up working on the same problem, when it could have been solved once by MIS. Data tends to be scattered throughout the company, making it hard to share and wasting space. Not following standards generates incompatibilities among systems, making it difficult to combine systems created by different departments or even by people within the same department.

End users are limited by the capabilities of commercial software. The initial systems may work fine, but as the company grows and changes, the commercial software might be unable to support the necessary changes. As a result, some users have created systems that produce incorrect answers, take too long to run, or lose data.

The last, and possibly most important, complication is that end-user development takes time away from the user's job. Some users spend months creating and modifying systems that might have been created by MIS programmers in a fraction of the time. One of the reasons for creating an MIS department is to gain efficiency from using specialists. If users are spending too much time creating and revising their own applications, the company needs to consider hiring more MIS personnel.

Development Summary

As a manager, one of the more difficult IT decisions you make is the choice of development methodology. As a business manager in a large organization, you might not have a vote. But within a smaller company, you will certainly have to look at the alternatives to help identify the most efficient means of creating projects. Even within a larger company, you might be in a position to suggest alternatives when price tags or time frames get too high.

Figure 12.22 summarizes the characteristics of the primary development methodologies. The chart is basically organized on a scale of formality. Large, formal projects are built using SDLC to control the development and record progress. Small scale reports and analyses can be created with end-user development. Prototyping is similar to end-user development, but is a step toward more control and formality because it uses trained MIS developers who follow established procedures and internal standards.

Remember that the various methodologies can be combined. For example, a JAD session might be used to define the initial goals and attributes of a large project. The forms might be refined through prototyping. But the overall project could be controlled through an SDLC project management system. Remember that each technology has different costs. SDLC provides the most control, but adds overhead costs that you have to recognize. On the other hand, prototyping

	SDLC	RAD	XP	JAD	Prototyping	End User
Control	Formal	MIS	Time	Joint	User	User
Time frame	Long	Short	Short	Medium	Short	Short
Users	Many	Few	Few	Few	One or two	One
MIS Staff	Many	Few	Many	Few	One or two	None
Trans/DSS	Trans.	Both	Both/DSS	DSS	DSS	DSS
Interface	Minimal	Minimal	Good	Crucial	Crucial	Crucial
Documentation, Training	Good	Limited	Variable	Limited	Weak	None
Integrity, Security	Vital	Vital	Unknown	Limited	Weak	Weak
Reusability	Limited	Some	Maybe	Limited	Weak	None

Figure 12.22

Comparison of methodologies. Each methodology has different strengths and weaknesses. You need to understand these differences so that you can choose the right tool for each project. Note that you can combine methodologies on large projects. For example, you could use prototyping to develop initial forms and reports that are incorporated into a larger SDLC project.

might appear to be inexpensive, but the costs could skyrocket if the project is never completed or requires huge amounts of developer and management time. One key issue in modern development is to identify these possible risks and threats up front. Then, each day, managers should evaluate the risks and see if the project has headed in the wrong direction. It might be impossible to prevent all risks, but at least if you are alert to the symptoms and recognize the problem earlier, you can correct it before the costs escalate and kill the project.

Process Analysis

How do you analyze and annotate a process-based system? If you are examining a transaction-processing system or dealing with a system that is largely noncomputerized, you should consider creating a process diagram. The purpose of a process diagram is to describe how the individual processes interact with each other. It concentrates on the business activities instead of the objects.

A data flow diagram is a process-oriented technique used for investigating information systems. The method can be used to look at the “big picture” and see how a system works in total. It also can be used to examine the details that occur within each process. Examining organizations at both levels provides a relatively complete picture of the problems and potential solutions. The use of systems analysis is illustrated by evaluating a small system for a zoo.

Input, Process, Output

One useful approach to systems is to look at them as a collection of processes or activities. The most important step in solving problems is to find the cause of the problems. Identifying the major processes in a system will help you under-

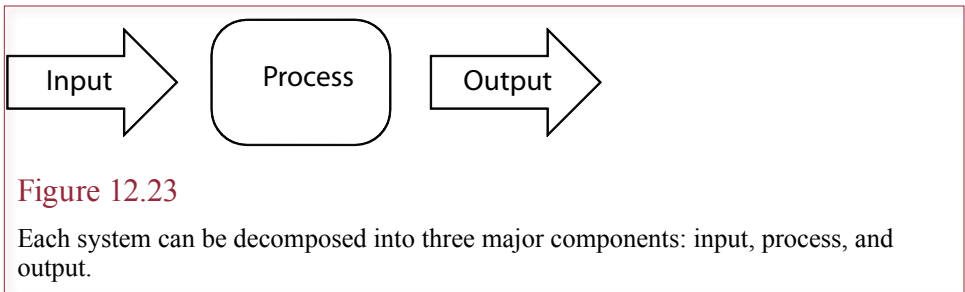


Figure 12.23

Each system can be decomposed into three major components: input, process, and output.

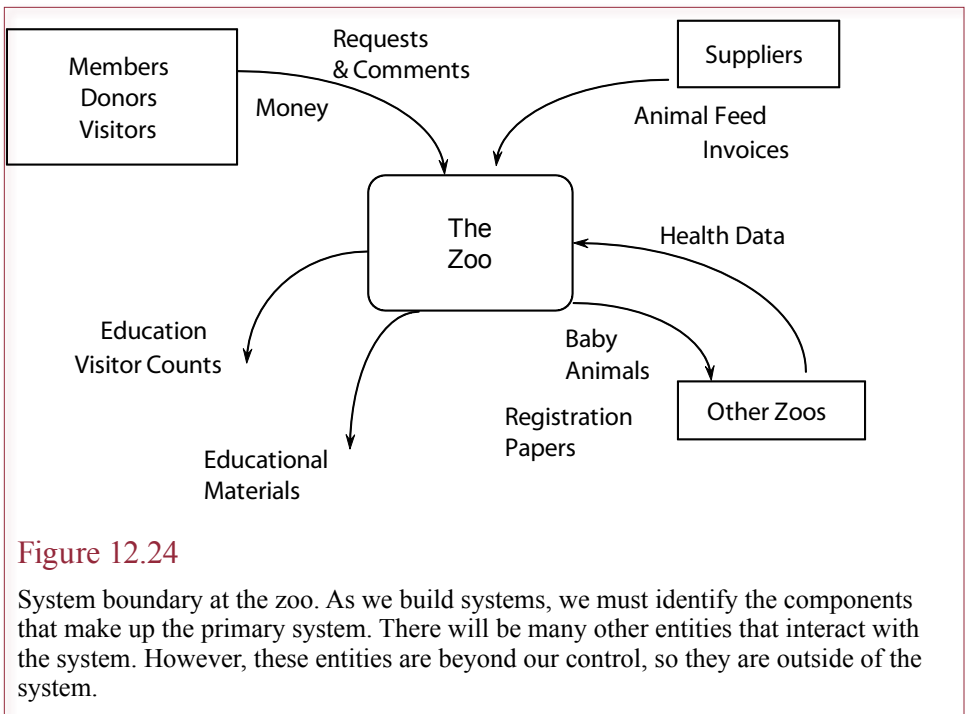
stand how the system works. Examining input and output objects helps you spot problems and trace them back to their source. As illustrated in Figure 12.23, systems receive *input*, which is *processed* to produce *output*. The process could be mechanical, such as manufacturing using raw materials, workers, and power. Alternatively, it might be a process involving symbolic processing instead of physical activity. For example, accounting systems receive sales data and process it into cash-flow statements. In many cases, there are two types of input and output: physical and data. Physical flows are often accompanied by data. For instance, raw materials are shipped with an invoice that describes the products and the shipping information. Systems theory can be used to examine both types of flow. However, this is an MIS text, so most of the problems presented here deal with flows of data.

Systems are described by collections of these processes. Each system operates in an environment that is somewhat arbitrarily defined by the boundaries of the system. For most problems, anything directly controlled by the firm is considered part of the relevant system. Everything else exists in the environment outside of the firm. The environment typically includes at least the physical space, laws, customs, industry, society, and country in which the firm operates. The firm can influence the physical environment, laws, and customs, but it does not have direct control over them.

Consider the example of a zoo: input and output are less concrete because a zoo primarily produces services instead of products. Figure 12.24 shows the basic inputs of food, money, and health data for potential new animals. Output objects include education, educational materials, and baby animals for other zoos. For most purposes, the system boundary is relatively clear. Visitors, suppliers, and other zoos are outside the direct control of the zoo, so they are in the environment. If the zoo was operated by a governmental agency, it would be harder to identify the boundary. Government systems tend to reach into many different areas, and it can be hard to identify their exact limits, especially since they can be extended or contracted by political decisions.

If a system is entirely self-contained and does not respond to changes in the environment, it is called a closed system. An open system learns by altering itself as the environment changes. Systems are almost never completely closed because closed systems cannot survive for long. However, some systems (or companies) are more responsive to changes in the environment than others.

Most large firms face a certain amount of inertia. It is easier for these firms to keep operating the way they always have than to continually introduce changes. But if a firm becomes too static, it can no longer respond to changes in the environment. Much like the U.S. railroad companies in the 1960s, closed firms will lose ground to firms that are more open and responsive to the environment. Re-



member that a key component of strategy is to search the environment for potential advantages.

Divide and Conquer

Most problems are too complex and too large to deal with all at once. Even if you could remember all the details, it would be hard to see how everything was supposed to fit together. A crucial step in analyzing a system is to carefully break it into smaller pieces or a collection of subsystems. Each subsystem is separate from the others, but they are connected and interdependent.

Figure 12.25 shows the five primary subsystems within the zoo. Of course, there could be many possible subsystems for the zoo. The actual division depends on how the organization operates. Each subsystem is defined by identifying the input and output flows. How do you know how to divide a system into smaller parts? Fortunately, most complex systems have already been subdivided into different departments and tasks. Many companies are organized by business functions: accounting, finance, human resources, marketing, MIS, and production. Others are split into divisions according to the type of product.

Once you have determined the major components of the system, each subsystem can be divided into even smaller pieces. An accounting department, for example, might be split into management reporting, tax management, quarterly reporting, and internal auditing groups. Each of these areas might be split into even more levels of detail. At each step, the subsystems are defined by what they do (process), what inputs are used, and what outputs are produced.

There are some drawbacks to the divide-and-conquer approach. It is crucial that you understand how the components work together. If a project is split into small parts and given to independent teams, the teams might lose sight of the

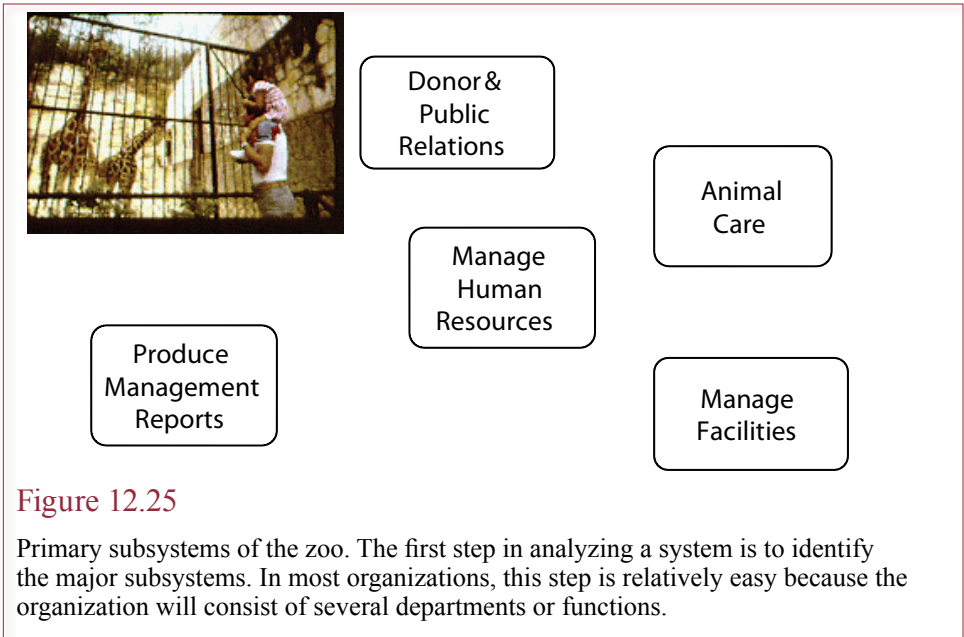


Figure 12.25

Primary subsystems of the zoo. The first step in analyzing a system is to identify the major subsystems. In most organizations, this step is relatively easy because the organization will consist of several departments or functions.

overall goals. Important components might not be completed, or the individual pieces might not meet the overall objectives of the system.

Goals and Objectives

Subsystems have goals or purposes. A goal of a manufacturing firm might be to sell more products than any rival (increasing sales). Or it might be to make as much money as possible for its owners (increasing revenues). Another goal might be to find an entirely new area in which to sell products (new market segments). The owners of the system should define its goals. If the system does not have a goal, it has no purpose and there is no way to evaluate it. In fact, by definition, it would not be a system. When you observe a system, you will need to evaluate performance, which means you have to identify the goals.

Typical spreadsheets give us the ability to ask “what-if?” questions. For example, you might want to know what happens if you increase sales commissions by 10 percent. Goals help focus the answer by providing the ability to ask *Why?* and *So what?* The answer to the *What-if?* question involving commissions might be that revenue increases by 5 percent. But what does that result mean? If we also know that a goal of the company is to increase profits, we could look more carefully and find that increasing commissions by 10 percent leads to a 3 percent increase in profits. That result is important because it brings the system closer to one of its goals. Hence, it would make sense to increase the commissions.

It is clear that to solve business problems, you must first identify the organization’s goals. The catch is that there are often conflicting ways to measure the goals. For instance, improved customer satisfaction or product quality might be useful goals. But how do we measure them? Managers who measure customer satisfaction by the number of complaints they receive will make different decisions than those who actively survey customers. In other words, the measurement of our performance with respect to the goals will depend on the data we collect.

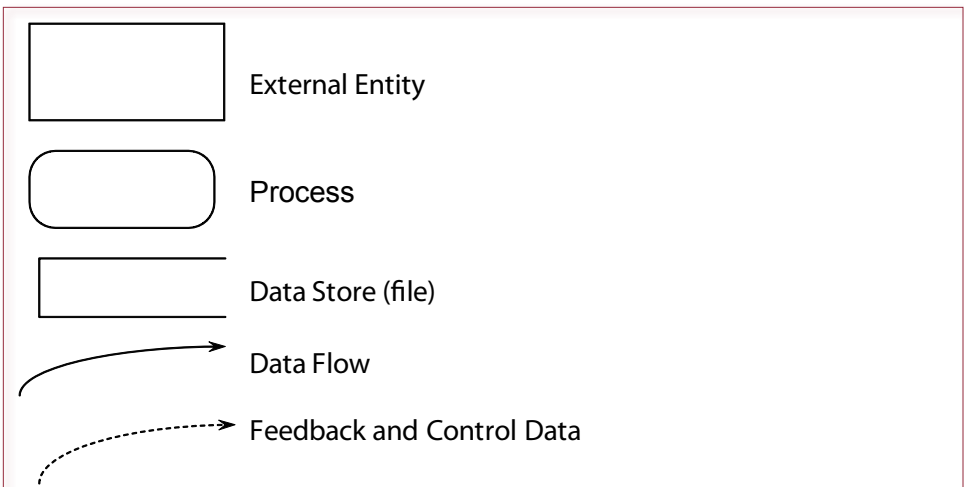


Figure 12.26

Only four or five objects are used to create a data flow diagram. External entities are objects that are independent and outside the system. Processes are functions and actions applied to data. A data store or file is a place to hold data. Data flows are shown as solid lines with arrows to indicate the data movement. Control flows are marked with dashed lines.

Diagramming Systems

We often represent systems graphically to gain insights and spot problems. We also use diagrams to communicate. Communication is of critical importance in MIS and all areas of business. Users describe their problems to systems analysts, who design improvements and describe them to programmers. Ideas and comments can be misinterpreted at any step. We can minimize problems by using a standard diagramming technique. The data flow diagram approach presented in this section is commonly used because it focuses on the logical components of the system and there are few rules to remember, so almost anyone can understand the diagrams.

Although you could invent your own diagramming technique, a method called a **data flow diagram (DFD)** has been developed to represent information systems. It is designed to show how a system is divided into smaller portions and to highlight the flow of data between those parts. Because there are only three graphical elements (five if you count the dashed control flows separately), it is an easy technique to learn. The DFD illustrates the systems topics in this chapter.

The basic elements of a DFD are external entities (objects), processes, data stores (files), and data flows that connect the other items. Each element is drawn differently, as shown in Figure 12.26. For example, data flows are shown as arrows. Feedback and control data are usually drawn as dashed lines to show that they have a special purpose.

Figure 12.27 presents the main level of subsystems for the zoo. Notice that it contains external entities, processes, and data flows. This level generally does not show data files or control flows. They can be incorporated in more detailed presentations.

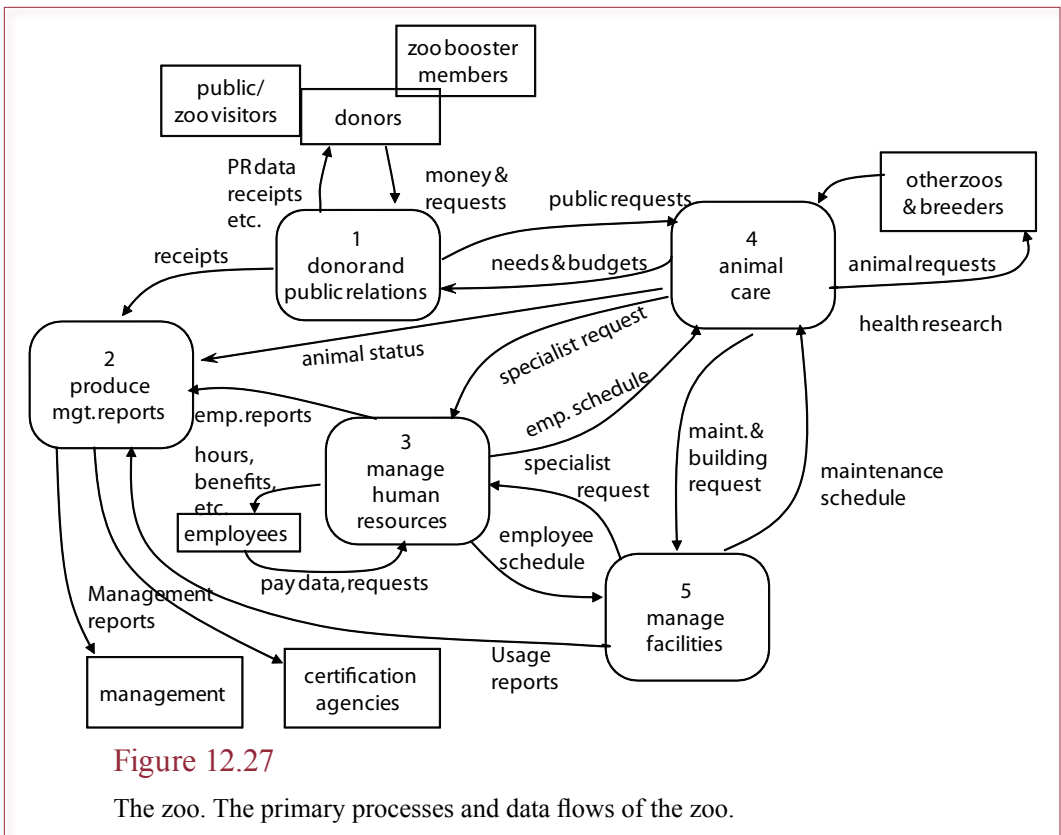


Figure 12.27

The zoo. The primary processes and data flows of the zoo.

External Entity

When you identify the boundary of a system, you will find some components in the environment that communicate with your system. They are called external entities. Although each situation is different, common examples include customers, suppliers, and management. External entities are objects, so they are labeled with nouns.

In the zoo example, the primary entities are management, certification agencies, other zoos, and members of the public (visitors, donors, and members). All relevant external entities need to be displayed on the first-level diagram.

Process

In a DFD, a process is an activity that involves data. Technically, DFDs are used to show systems that involve data, not products or other materials. However, in business today, virtually all physical processes have data-processing counterparts. When customers buy something, they get a receipt. In a manufacturing process, the amount of raw materials being put into a machine, measures of the volume of output, and quality control values are recorded. The DFD process is used to represent what happens to the data, not what occurs with the raw material.

Because processes represent actions, they are typically labeled with verbs, such as *sell products* or *create tax reports for management*. There are two important rules involving processes. First, a process cannot invent data. That means every process must have at least one flow of data entering it. Second, a process cannot be a black hole; every process must transfer data somewhere else. If you

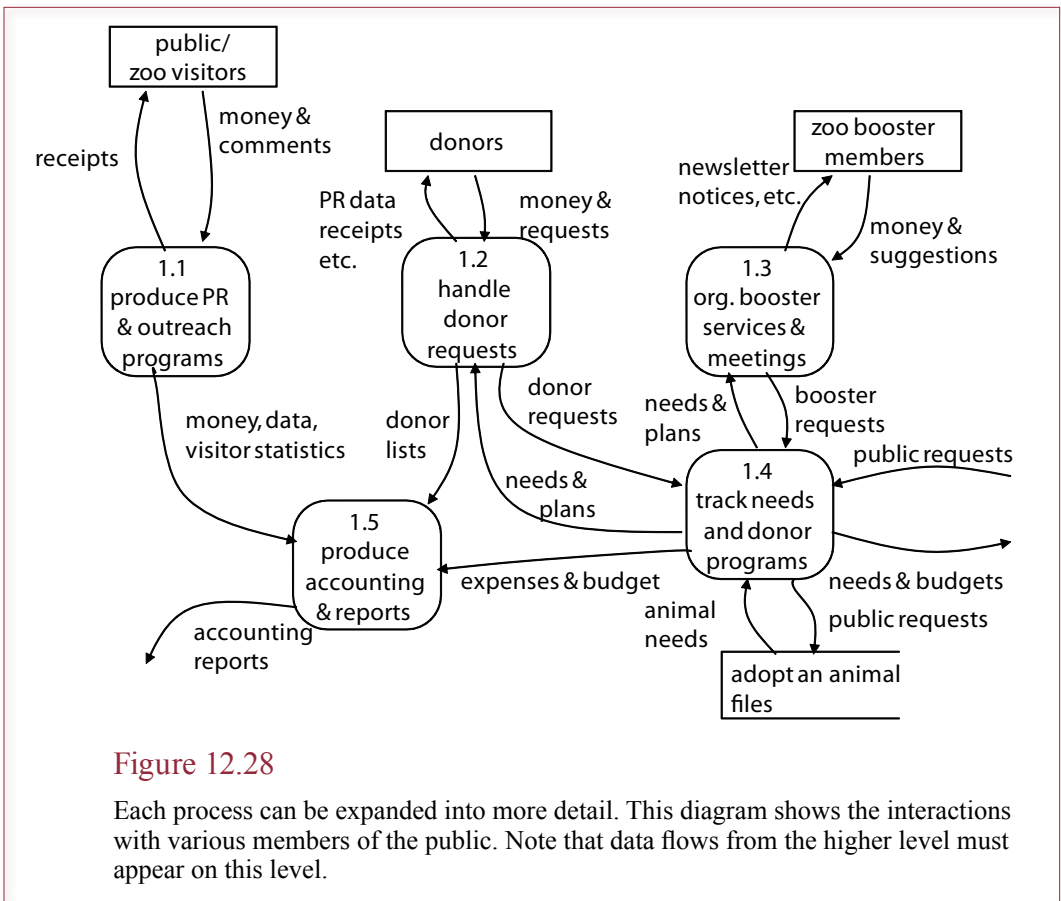


Figure 12.28

Each process can be expanded into more detail. This diagram shows the interactions with various members of the public. Note that data flows from the higher level must appear on this level.

look at your DFD and find one of these two problems, it usually means that you missed a connection between the processes. On the other hand, processes that do not export data might be data stores or external entities.

Data Store

A data store or file is simply a place to hold data for a length of time. It might be a filing cabinet, reference book, or computer file. In a computerized system, data is likely to be stored in a database management system (DBMS). Chapter 5 provides more detail on the capabilities and uses of a DBMS. For now, it is important to note that data is a valuable resource to any company. In drawing a DFD, try to list exactly what needs to be stored, how long it should be held, and who should be able to read or change the data.

Data Flow

The data flows represent the inputs and outputs of each process or subsystem. The data flows are easy to draw. They are simply arrows that connect processes, entities, and data stores. Be sure to label every data flow. The diagram might seem obvious *now*; however, if someone else reads it or you put it away for several months, it can be hard to figure out what each flow represents.

Processes	Description . . .
Animal care	Feed, clean, and vet care
Donor & public relations	Handle public requests and provide educational information
Employee relations	Schedule employees, process benefits, handle government reports
Facility management	Handle maintenance, new construction, planning
Produce mgt. reports	Collect data and produce summary reports for management
Entities	
Certification agencies	Government and private agencies that create rules and regulate zoos
Donors	People and companies who donate money to the zoo
Employees	Primary (paid) workers, full-time and part-time
Other zoos and breeders	Zoos we trade with and share data with
Public/zoo visitors	Daily visits, we rarely keep data on individuals
Zoo booster members	Members who donate money and time for minor benefits
Data	
Accounting reports	Standard (GAAS) accounting reports for management
Certification reports	Reports for certification agencies; produced annually
Facility reports	Summaries of work done and plans, mostly weekly
Needs and budgets	Budgets and special requests from animal care
Public requests	Suggestions and comments from the public

Figure 12.29

A few sample entries from the zoo's data dictionary. A data dictionary records details on all of the organization's objects. It is typically organized by type of object. It is easiest to maintain if it is stored in a computer database.

Division of the System

A DFD provides an excellent way to represent a system divided into smaller components. First, each task is shown as a separate process. The data flows between the processes represent the inputs and outputs of each subsystem. Second, the DFD for a complex system would be too large to fit on one page. Hence, the DFD is displayed on different pages or levels. The top level, or *context diagram*, acts as a title page and displays the boundaries of the system and the external entities that interact with the system. The next level (level zero) shows the primary subsystems. Figure 12.26 is an example of a level zero diagram. Each of these processes is then exploded into another level that shows more detail. Figure 12.28 is the exploded detail for the first process (donor and public relations). These explosions can continue to any depth until you have displayed all the detailed operations needed to explain the system.

Data Dictionary

In any project, you need to remember additional pieces of information about each object. You might want to keep a sample report for a management tax report data flow, along with any deadlines that must be met. For data stores, you need to record information such as who controls it, who needs access to the data, how often it should be backed up, and what elements it contains.

A **data dictionary**, or repository, contains all of the information that explains the terms you used to describe your system. A good computer-aided software engineering (CASE) tool will maintain the dictionary automatically and help you enter longer descriptions for each item. Without these tools, you will have to keep a notebook that contains the full descriptions. For convenience, the entries should be sorted alphabetically. A word processor can be used to hold and print the dictionary. Figure 12.29 shows sample entries for the zoo system.

Summary: How Do You Create a DFD?

The first step in creating a DFD is to identify the environment and boundaries of the system by asking the following questions: What problems do you need to solve? What areas do you want to avoid? What are the goals? What are the main external entities? The second step consists of identifying the primary processes that define the system. Keep the list short (fewer than 10). Then answer these questions: What are the main activities in the system? What are the inputs and outputs of each process? How are these processes interconnected by the data flows? The third step is to look at each process in more detail and draw an expanded subsystem on a new page. What activities take place within a given process? What detail is needed in the reports and data inputs? The fourth step is to build the control flows. What processes are used to monitor progress toward the goals? What additional data is collected to monitor the environment and the system's performance?

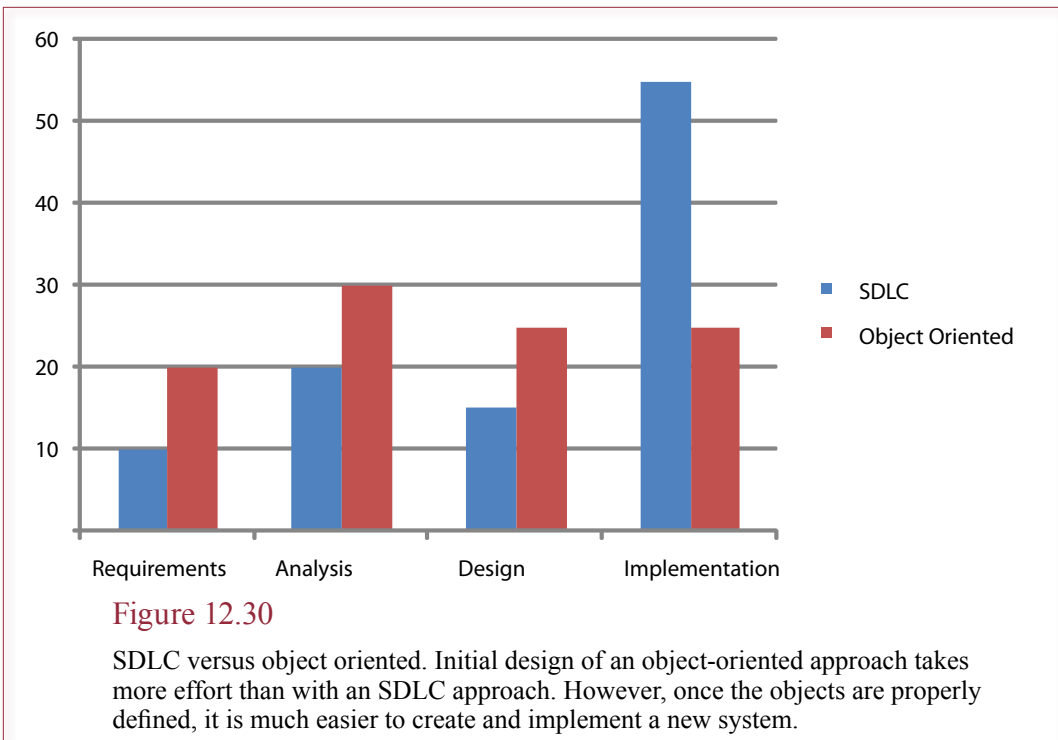
The key to analyzing systems is to start small. You can begin with one detailed subsystem and build your way up, or you can describe the general system processes and work down by adding increasing levels of detail.

Object-Oriented Design

How is object-oriented design different from process design?

One way to begin your analysis of a business is to focus on the business objects: what they are and what they do. Objects could be anything from people to raw materials to data files or schedules. The key to object-oriented design is to focus on defining what an object is and what it can do. A class is a generic description of a set of objects. This distinction is not crucial in this book, but you might want to know there is a difference. For example, the Bicycle class describes any bicycle that could be built by the company. A specific bicycle (e.g., serial number 15) is an object.

The concept of object-oriented design has received considerable attention during the past few years. In some ways, the base design techniques are not much different from traditional SDLC techniques. In other ways, object orientation requires a completely new way of thinking about systems development. The ultimate goal of the object-oriented approach is to build a set of reusable objects and procedures. The idea is that eventually, it should be possible to create new systems or modify old ones simply by plugging in a new module or modifying an existing object.

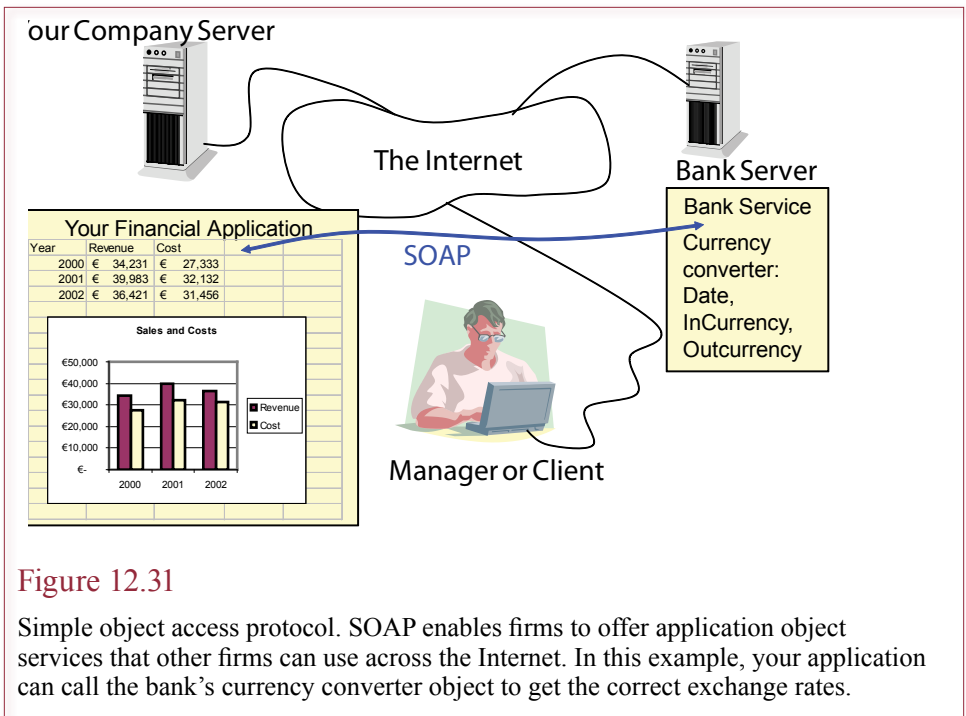


One key difference between object orientation and other development methods is the way processes or functions are handled. With objects, all functions are embedded in the definition of the object—the object comes first. The object approach reverses the treatment of processes and data. With SDLC, illustrated by a data flow diagram, the emphasis is on processes, and data (attributes) is passed between processes.

One goal of an object-oriented approach is to create a set of information system building blocks. These objects and procedures could be purchased from commercial software companies (such as a spreadsheet from Microsoft or a database system from Oracle). MIS programmers or consultants can create additional objects tailored for your specific company or department. Once the basic blocks are in place, end users or MIS analysts can select the individual pieces to create a complete system. Hence, as Figure 12.30 indicates, less time is needed for implementation, as long as the analysis and design are performed carefully. On the other hand, the up-front costs of designing and building these objects can be quite high. Furthermore, the tools and techniques tend to require substantial retraining of the existing MIS staff. Both of these types of costs have caused some companies to avoid object-oriented methods.

Distributed Services

Can software be located in multiple places? A major question in decentralization is where the software needs to be located. Substantial benefits arise from centralizing data—providing access across an intranet or the Internet. But users need some type of distributed hardware to access the data. Does that mean all of the software has to be installed on each machine? Some new technolo-



gies are being developed that provide support for distributing software functions across the network. While basic tools such as a word processor are still needed on individual machines, the complex business and analytical tools can be installed on central servers.

One of the primary technologies is the **simple object access protocol (SOAP)**. It is a standard being pushed by several vendors to define how objects can be used across the Internet. It relies heavily on the **extensible markup language (XML)** to transfer data between diverse computers. As a general manager, you do not need to know the details of how these two technologies work, but you should remember their purpose. Ultimately, you will want to select applications that fully support these standards so that you can build and use systems that work transparently across the Internet.

The purpose of SOAP is to enable firms to build application services that can be used by other organizations across the Internet. For instance, as shown in Figure 12.31, a bank (e.g., www.oanda.com) might offer a currency conversion application. Your company's accounting application could call the bank's program whenever it needed to convert money to a different currency.

Applications that use the SOAP and XML protocols can interact with other services across the Internet. However, a big question that remains to be resolved is how firms will price their services. Firms that create service objects will ultimately be able to bill clients a usage fee or a monthly charge with unlimited access. But a standardized billing mechanism has not been implemented yet.

Cloud Computing

How does cloud computing change software development? In one respect, cloud computing has made computing easier for companies. Essen-

tially, cloud computing offers **software as a service (SaaS)**, where a few firms hire developers and create and host software and other firms simply lease the service. That means that only a few firms develop software. This trend has been evolving for several years—partly because new technologies make it feasible, but partly because most companies do not want to spend money and time trying to create custom software and pay large groups of programmers. The economics of software development also explain cloud computing because the high development costs are spread across many other companies. Still, not all companies are moving to publicly-run cloud computing. Issues of privacy and customization have held back some of the acceptance. Instead, smaller firms opt to buy software and install it on their central computers—and larger companies might run their own private clouds.

The second issue of cloud computing consists of developing software that runs on the cloud. First, most of the software relies on Internet technologies and databases. Initial technologies (Web browsers) provided limited support for interaction and usability. But, the Internet technologies continue to evolve, which provides more capabilities but requires learning and testing new tools and new designs. For instance, HTML 5 was introduced by Web browsers in 2011, with advanced interaction features, but the standard and full support are not expected for at least three years. Consequently, developing software for cloud computing remains somewhat experimental. Any experimental technology is more difficult to predict, and it is harder to estimate development time and cost. Also, it is harder to develop software that is designed to be used by multiple companies. Additional security elements need to be designed and tested, and the system has to be built to be scaled up as more companies use the products. Scalability and expansion are key elements in designing any cloud-based system. Not just the hardware, but the software and the administration tools have to be designed so they can be managed without significant increases in the number of administrators. Effectively, most cloud-based software carries additional development costs because administration tools need to be built into the system.

Summary

Systems development can be a difficult task. Many projects have failed because they cost much more than anticipated or they did not produce useful systems. Large projects are especially difficult to control because there can be conflicting goals, it is hard to ensure that subsystems work together, business needs change during the development process, and there is turnover among the MIS employees. The systems development life cycle evolved as a means to deal with the complexity of large systems and provide the necessary controls to keep projects on track.

Systems analysis techniques are used to break projects into manageable pieces. Various graphing tools, such as data flow diagrams, are used to display the relationships between the components. Systems design techniques use the results of the analysis to create the new system. The new system consists of interconnected modules. Each module has inputs, outputs, processing steps, database requirements, manual procedures, and controls. At various stages in the design process, managers and users are asked to sign off on the proposed system, indicating that they will accept it with no further changes.

In contrast to the rigid control embodied in the SDLC method, the prototyping approach is iterative and creates an early working model of the system. Users and managers can see the proposed input screens and reports and make changes

to them. As the project develops, the prototype goes from a simple mockup to a working system. Prototyping is sometimes used in conjunction with SDLC during the design phase to lay out input screens and reports.

A third way to build systems is for end users to develop their own projects using fourth-generation tools such as database management systems, spreadsheets, and other commercial software. As the capabilities of commercial software tools increase, users can develop more complex systems. The backlog facing MIS also encourages users to develop their own systems. The potential dangers of user development, such as lack of testing, incompatibilities, and unnecessary duplication, can be controlled by having MIS teams available to assist end users.

All methods of developing systems involve five basic steps: feasibility and planning, systems analysis, design, implementation, and maintenance. Prototyping and end-user development typically focus on the design stage. However, managers need to remember that implementation problems can arise with any new system, regardless of how it was created. Similarly, there will always be a need to maintain and modify existing applications. It is easy to forget these steps when users develop their own software.

A Manager's View

As a manager in a large company, you will work closely with the MIS department to modify and build systems that support your operations. You need to be aware of the problems facing MIS staff to understand the reasons for their rules and methods. Managers are increasingly being asked to develop their own systems and to participate more heavily in the design of new reports and forms. The details of analysis, design, testing, and implementation will be useful regardless of the method used. As a manager, you also need to know the advantages and drawbacks of various development methods; you will often have to choose the method that is best suited to solving your problems.

Key Words

best practices	object-oriented programming (OOP)
capability maturity model integration (CMMI)	open source development
change agents	outsourcing
commercial off-the-shelf software (COTS)	phased implementation
data dictionary	program logic
data flow diagram (DFD)	prototyping
end-user development	rapid application development (RAD)
event-driven	reusability
extensible markup language (XML)	scope creep
extreme programming (XP)	simple object access protocol (SOAP)
fault tolerance	software as a service (SaaS)
feasibility study	software maintenance
inheritance	syntax
joint application design (JAD)	systems analysis
object hierarchy	systems analyst
	systems development
	life cycle (SDLC)



Web Site References

	Computer Industry News
ACM Digital Library	portal.acm.org/dl.cfm
CNet News	news.com.com
Computer Economics	www.computereconomics.com
Computerworld	www.computerworld.com
eWeek	www.eweek.com
Federal Computer Weekly	www.fcw.com
Gartner Group	www.gartner.com
Government Computer News	www.gcn.com
IDG	www.idg.com
IEEE	www.ieee.org
Infoworld	www.infoworld.com
Internet.com	www.internet.com
PC World	www.pcworld.com
The Industry Standard	www.thestandard.com


Review Questions




1. What fundamental methods are available to build information systems?
2. What are the main elements of programming logic?
3. How does object inheritance simplify programming?
4. Explain why the first step in most business projects should never be programming.
5. What is the primary purpose of the systems development life cycle development methodology?

6. What are the main steps in the systems development life cycle methodology?
7. What drawbacks are created with the systems development life cycle methodology?
-  8. What alternative methods are being used to develop information systems?
9. How does agile or rapid application development speed up the development process?
-  10. What is the role of a data flow diagram in analyzing systems?
11. What are the main components of a data flow diagram?

Exercises

1. Interview a local manager to determine the requirements for a new system. Explain which method would be the best approach to develop the system. Estimate how long it would take to complete the project and how much it would cost. Advanced option: Illustrate the new system with a data flow or object-oriented diagram. More advanced: Create the system.
2. Create a Web page form with 5 text boxes and add Javascript code to check that each box is not empty when the form is submitted.
-  3. Find a small example of a business or Web program, choose a function or subroutine and explain its purpose.
4. A regional bank office generates loans for builders. The office has several bankers who form alliances with regional builders and negotiate loans and other services. The manager wants a system to track the leads, including the potential amount of the loan and the probability of the loan going through. Every month, the main office sends a spreadsheet file with current loan information. The manager and the main office want the regional bank officers to project the amount of money that will be loaned in the coming months. The manager wants a system to help collect and track this data. Identify the best development methodology. Assuming no one in the regional office has the skills to create the application, do some research to find at least two firms that could handle the job for a reasonable price.
5. A large lawn-maintenance company wants a new system that uses smart phone applications to collect data for its operations. The company has ten trucks and 20-30 people who mow lawns and other yard work for businesses and homes in a large city. To help reduce travel time and gas costs, the company wants to use GPS to track the daily routes of the 30 trucks and then later optimize the routes. It also wants to use this information to contact the nearest truck when a customer has a special request, or to direct the nearest truck to a new client who is requesting a quote. The phone application also needs a screen for employees to enter the time they spend on each project, and other screens to handle purchases of items such as gas, fertilizer, and tools. Maintenance for each of the mowers and other power tools should also be handled on the smart phone. Ideally, bar codes will be placed on each piece of equipment and the phone's camera can be used to scan the code to identify the equipment. The system also needs to support communication

between the workers and the clients. For instance, clients should be able to enter special requests, and workers can enter problems, such as broken sprinklers. Create a design for this application.

6. For each of the following information system projects, identify the development method that would be the best approach for most companies.
 - a. An electronic system to handle travel requests and data entry for reimbursements.
 - b. A smart phone app that is a cross between a map and a social network, where people enter locations of products they find in stores so other people can follow the map. (Where are the pickles in a Safeway store?)
 - c. An application to use traffic cameras to track the location of specific cars using vision-recognition tools to follow license plates.
 - d. A program to track health and feeding data at a zoo (for animals not visitors).
 - e. An application for a couple of people in marketing to track viewings of radio and TV ads.
 - f. Modifying an ERP system by adding forms, rules, and reports to track expenses related to a new product design team created from an acquisition.
7.  Assume that you are on a project to build a new Web site for a midsize company. The firm sells materials to home builders—usually contractors, but some individual sales as well. The company wants to take orders over the Web and enable customers to track the status of current orders. Contractors also want the ability to look at old orders when placing new ones. For example, if they build the same style of house twice, they will need approximately the same materials. The manufacturer is not completely certain on many of the details yet, and you will have to interview customers to get additional details and feedback. Set up a schedule for developing this system using the SDLC approach. Then, identify ways that RAD might be used to reduce the overall development time.
8. You work for a company that is increasingly asking employees to develop their own applications using Microsoft Office tools connected to the corporate database. This process has not been working very well, and employees are grumbling. But the company has decided it cannot afford to hire all of the MIS people that would be needed to develop all applications and reports within the MIS department. How can the company improve the process? What tools and capabilities should the company add?
9. Check out www.sourceforge.net. Briefly explain its purpose. Find two programs that might be useful to businesses.
10. After several decades of challenges building systems, improving tools, and new development methods, why are many IT projects still over budget and late?



Technology Toolbox

11. Write a short macro program in Excel that adds all of the numbers between the values in cell A1 and cell A2 and puts the result in cell A5. For example, if A1 = 1 and A2 = 5, then add $1 + 2 + 3 + 4 + 5$ to get 15. *Hint:* You can read or write to a cell with the command Range("A1").
12. Write an Excel macro that looks at each item selected to see if any cells are blank. If any are blank, display a message notifying the user how many blank cells there are. (*Hint:* Use the IsEmpty function to test and the MsgBox command to display a message.)
13. Create a form in Microsoft Access (or Visual Studio). Place text boxes on the form for amount to borrow, interest rate, and number of months. Add a fourth text box to hold the resulting payment amount. Set the properties to format and name each of the boxes. Add a button to calculate and display the payment amount based on the entered data. Use the Pmt function to do the calculation.
14. Use InfoPath to create and publish the expense report form. If possible, save it to a SharePoint server. Enter sample data and save and submit the form. E-mail it to your instructor as your supervisor.
15. Talk with a manager or employee to identify internal forms and data that are collected. Make a list of at least 5 forms for a business that could benefit by using InfoPath.
16. Find a tool on the Web that could be used to create forms similar to InfoPath (but probably simpler), but runs on a Web browser and stores all the data on a Web site.



Teamwork

17. Interview computer users to determine how they feel about their current system. Do they like it? What are the major advantages and drawbacks? How long have they used it? When was it changed last? Are there changes users want to see? Are they willing to accept changes? How are relations with the MIS workers? Who initiates changes, users or MIS? If users proposed a new project, how long would it take for MIS to get to it (how long is the backlog)? Each team member should interview a different person (some users, some in MIS). Combine your results to get a picture of the entire company. Do users agree with each other? Does the MIS department agree with the users? Do they see the same problems? (*Hint:* If you do not have access to another company, you can always find computer users in the university.)
18. Choose one person in the team who has an interesting job. Create a data flow diagram for the job and organization. Be sure to label everything and provide a data dictionary.
19. Find a manager who needs a computer project completed or updated. Interview the person and record the comments and notes. Create a basic design for the system. Identify the best methodology to create the system. If possible, find someone to build the system.

20. Rolling Thunder Bicycles wants to create a new Web site to enable customers to build and order bicycles online. It should also enable customers to view the progress of the construction and make payments online. Assign a different development methodology to each person who will perform a basic design and argue why that method should be used for the project. Select one of the methodologies in the end.
21. Examine the sample pseudocode used to compute the total of a set of numbers. Choose at least two programming languages and assign team members to a specific language and find or write the code needed to compute the total in that language. Submit and comment on the differences in the languages. *Hint: The code does not need to run and you can skimp on the file and print statements.*
22. Create a simple form using InfoPath that collects at least three pieces of data such as name and e-mail address. Test the form by sending it to each person on the team. Collect the data in a spreadsheet or simple database table.
23. As an exercise in creativity, each person should write down a business or personal task that they would like to see computerized or available on smart phones. Consolidate the list and search the Web to see what tools already exist to handle the task.



Rolling Thunder Database

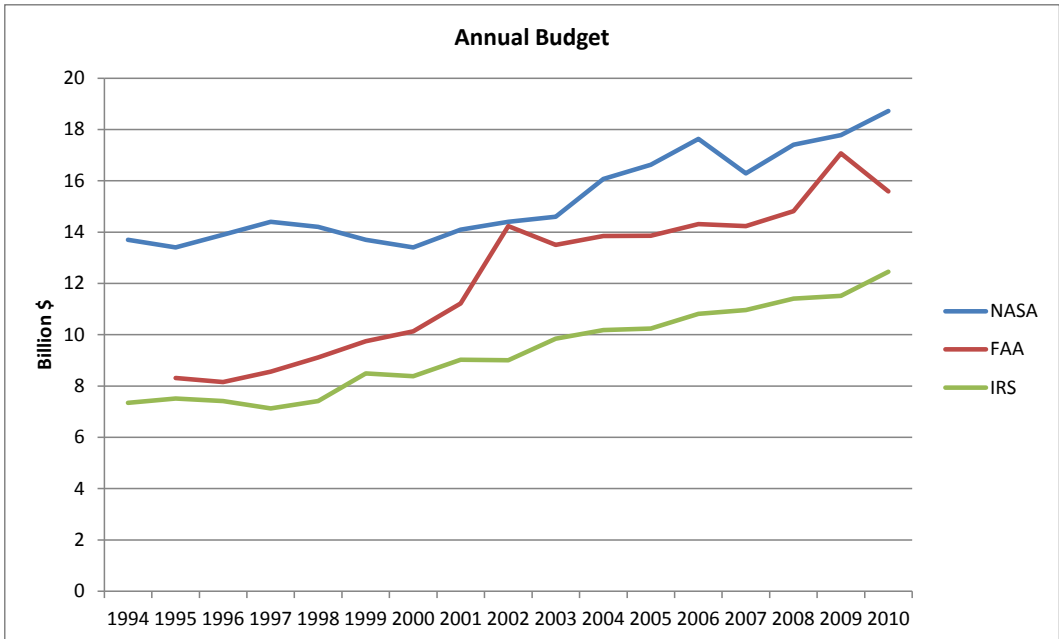


24. Rolling Thunder bicycles needs a new Web site to sell its custom bicycles. How should it be developed? What methodology could be used?
25. Using the help system and Web site description of Rolling Thunder, create a data flow diagram to show the main processes directly involved with the customers (taking orders, sending notices and bills, and receiving payments).
26. Rolling Thunder Bicycles needs a new system to generate and track electronic orders (EDI) to its suppliers. What methodology should be used to develop the system?
27. Assume that the managers of Rolling Thunder bicycles have decided to purchase and implement an enterprise resource planning system. You have been selected to help determine which system the company should purchase. Outline the steps you will have to perform to select a vendor.
28. Identify at least three information processing tasks at Rolling Thunder Bicycles that could be handled with InfoPath. Compare the benefits and drawbacks of using InfoPath versus adding new forms to the existing application.

Additional Reading

- Boehm, B.W., Software Risk Management Principles and Practices, *IEEE Software*, 8(1), 32-41. [Top 10 risk items in development.]
- Brooks, Frederick P. *The Mythical Man-Month, Anniversary Edition*, 1995, Boston: Addison-Wesley. [Classic book on software development problems and why adding people to a project often slows it down.]
- DeMarco, T. and T. Lister, *Peopleware*, New York: Dorset House, 1987. [Hints and problems developing useful systems.]
- Jones, T.C., *Programming Productivity*, New York: McGraw Hill, 1986. [Evaluating and measuring productivity, costs, and teamwork.]
- Jeffries, Ronald, Ann Anderson, Chet Hendrickson, *Extreme Programming Installed*, 2001, Boston: Addison-Wesley. [A detailed explanation of how to use extreme programming. Also look at other books in the series for theory.]
- Naumann, Justus and Milton Jenkins, “Prototyping: The New Paradigm for Systems Development,” *MIS Quarterly*, Spring 1982. [Description, uses and advantages of prototyping.]
- McConnell, Steve. *Rapid Development: Taming Wild Software Schedules*, Redmond, Microsoft Press, 1996.
- Paulk, Mark C., Bill Curtis, Mary Beth Chrissis, and Charles V. Weber, *Capability Maturity Model for Software*, 1993, Carnegie-Mellon University, Software Engineering Institute. <http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr24.93.pdf> [Description of how to evaluate software development programs in terms of their capabilities on a relatively standardized maturity scale—essentially an evaluation of how well an organization follows the SDLC.]
- Wallnau, Kurt, Scott Hissam, and Robert C. Seacord, *Building Systems from Commercial Components*, 2002, Boston: Addison-Wesley. [Description and cases of developing systems using commercial off-the-shelf software components, from the Carnegie Mellon Software Engineering Institute.]

Cases: Government Agencies



Most U.S. citizens know the overall structure of the federal government: the president, Congress, and the Supreme Court. These groups are responsible for creating and interpreting the laws to govern the nation. What many people do not realize is that both the president and Congress are supported by a huge set of government agencies. These organizations form a bureaucracy that is ultimately responsible for carrying out the laws. Governmental agencies have several unique problems. The most important one is that funding is subject to changes in the political climate. With each election, an agency runs the risk of having to change direction, cancel projects, or provide support for new tasks.

On the other hand, most government agencies are not subject to economic pressures. Consequently, they have not been faced with the same incentives to economize and minimize costs that have faced businesses. Another critical feature of most government agencies is that they tend to serve large numbers of people, especially at the federal level. These large organizations collect huge amounts of data. Increasingly, these agencies are converting to electronic storage and access. Publicly available data can often be found on Web sites—at least at the federal level.

Most governmental agencies have dealt with the size issue by maintaining large staffs, and combining decentralized management with centralized controls. Traditionally, government organizations have paid lower salaries than commercial businesses. Although the salaries are supplemented with benefits and job security, governmental agencies often face high turnover rates and changes in personnel. To compensate for these problems, the agencies rely heavily on procedures. These rules seek to predict and then direct what to do in circumstances that may arise. As new situations and decisions present themselves, new rules are created. Given these challenges, there is no surprise that most people perceive government agencies as large bureaucracies, filled with endless forms and strange rules.

There are many obvious uses for computers in government agencies. During a few minutes of observation, anyone can generate ideas that could improve agency performance, making life easier for the workers and citizens. However, the real challenges have always come in creating and implementing these ideas. In the 2000s, the number of students graduating with degrees in computer science and information systems plummeted. As demand for technical skills began to increase in 2010, many companies found it difficult to find qualified new employees. This problem was even worse for government agencies—many of the top-notch, innovative programmers do not want to work for government agencies. Plus, many of the graduates of tech programs are not U.S. citizens, so they cannot get federal jobs (Piemonte 2011).

Although many success stories exist regarding computer implementation within government agencies, there are also some costly failures. The Federal Aviation Administration and the Internal Revenue Service cases present some of the difficulties that have arisen.

Be careful when you read these cases. Do not simply blame the problems on “typical government mismanagement.” Many of these problems also exist within businesses. Always remember that the challenge is to search for and implement answers and methods that will overcome the obstacles and complications.

Size and Growth

The federal government employs 2 percent of the U.S. workforce, with about 2 million civilian employees in 2010, not counting the Postal Service. In 2003, it spent \$757 billion, rising to 808 billion in 2006 and to \$1.26 trillion in discretionary spending in 2010. With the tax cut of the early 2000s, federal receipts declined in 2002 and 2003, while expenditures continued to increase about 6 percent a year. The federal debt rose to 2.7 percent of GDP in 2003. Part of the economic balance is due to the war in Iraq; part is due to the attempts to prevent a major recession. In 2010, total federal government expenditures were \$3.456 trillion, with receipts of \$2.163 trillion, leaving an annual deficit of \$1.293 trillion. To put it in context, expenditures amount to almost 24 percent of gross domestic product (GDP) for 2010. And budget projections for future years showed increasing expenses and deficits (2012 Federal Budget).

In some ways, the size of the federal government is shrinking. The number of elected officials remains constant but the rest of the federal government is downsizing. Federal government employment peaked in 1990 with a total number of 3.233 million employees. This number decreased through 2007 and increased in 2008, 2009, and probably 2010. In 2009, the total was 2,804 million. On the other hand, several widely circulated reports from most federal agencies indicate that by 2008, over 50 percent of the federal workforce will be eligible for retirement. Not all of them will actually retire immediately, but over the course of a few years, a substantial percentage of the federal workforce will need to be replaced. A considerable amount of internal knowledge could potentially be lost in the process. The Census Bureau reports that in 2007 excluding Postal Service employees, the Federal Government employed 1.845 million full-time civilians. In 2008, that number was 1.885 million and it dropped to 1.789 million in 2009.

In comparison, for the United Kingdom in 2003, public spending amounted to 41.1 percent of its GDP. Deficits in European nations routinely run about 5 percent of GDP (*The Economist* 2004). However, remember that the UK pays for health care. In the United States, if you include state and local government spend-

ing, total government spending was over \$2 trillion in 2003 or 18.7 percent of GDP. If you include current healthcare spending, the total would be about 35 percent of U.S. GDP.

Information Technology

Like any business, government agencies increasingly rely on technology to improve productivity. Most agencies are under constant pressure to reduce costs—often to the point of having their funding cut. In large agencies, such as the IRS and the FAA, monster technology projects are funded separately. Consequently, the agencies have often been able to obtain funds specifically set aside to acquire or build new systems.

Most tasks performed by government agencies are unique. As a result, they require custom-developed software. Since the projects are huge and involve a large number of users, they are difficult to develop. Throw in a few bureaucratic turf battles, and it is amazing that anything gets done. Unfortunately, the result has been that many agencies are operating with technology that is 10, 20, or more years out of date. These archaic systems create their own ongoing problems. The government cannot just stop what it is doing, throw the old systems away, and build new ones. Instead, most of the government IT workers keep the old systems running. So, who is going to build the new systems? And how do you obtain the detailed knowledge from the users and workers of the old systems? Then, how do you transfer all of the data and keep both systems updated and running while testing the new system? However, we know that information technology constantly changes. Somehow, agencies have to balance the costs against the capabilities of the new systems.

In December 2010, Vivek Kundra, the U.S. CIO established a policy called “cloud first” which told federal agencies to move at least three services to cloud computing within 18 months. The goal was to decrease costs and improve flexibility. Some agencies found it relatively easy to comply; others were focused on evaluating operating costs, security, and bandwidth needs. For example, the agency responsible for the American Recovery and Reinvestment Act of 2009 used Amazon’s EC2 to run the Recovery.gov Web site for sharing data related to the Act and saved about \$750,000 in the first year [Pratt 2011]. Likewise, the U.S. Treasury Department moved the Treasury.gov site to Amazon cloud services. For more secure operations, the Defense Information Systems Agency (DISA) and NASA have created their own cloud servers (RACE and Nebula) that can be used to host operations from other agencies. The General Services Administration (GSA) is defining standard contracts and schedules to make it easier for agencies to purchase cloud services from commercial vendors. In the meantime, any classified data is unlikely to be moved to commercial clouds.

Oversight

Congress is charged with appropriating all federal money and controlling spending. Yet members face an enormous bureaucracy, plus constant reelection worries to distract them. Consequently, the General Accounting Office (GAO) was created to help monitor the spending and procedures at the various agencies. As a nonpartisan office, the GAO is free to collect whatever data it wants from the agencies and yell at them as necessary. You can obtain GAO reports directly from www.gao.gov. The reports often contain detailed information on specific projects and audits. The agencies generally provide responses to GAO concerns within the

report. These responses are always interesting because the agency director tiptoes a line. No one wants to accept all of the GAO criticisms (and look like there is no control), yet no one wants to totally disprove the GAO, because that would mean no more funding is needed.

Additional Reading

Bureau of Economic Analysis, <http://www.bea.gov/>

The Economist, “Has Tony Wasted Your Money?” July 10, 2004, p. 12.

Piemonte, Phil, “Many Grads, But Not the Right Stuff,” *Government Computer News*, June 21, 2011.

Pratt, Mary K., “Feds Race to the Cloud,” *Computerworld*, July 13, 2011.

Statistical Abstract of the United States, <http://www.census.gov/compendia/statab/>

Case: Federal Aviation Administration (FAA)

The FAA is charged with overseeing all public (nonmilitary) flight operations in the United States related to safety and access to the air. They establish safety criteria, issue licenses for pilots, and create air worthiness certificates for planes. They also operate the air traffic control system throughout the United States. Funding for the agency is generated through user fees and taxes on aircraft fuel, tires, and airline tickets. The FAA is an executive agency that operates under the budgetary control of the president. Appropriations and organizational structure for the agency are approved by the Congress.

The increase in air traffic in the United States has made air traffic control a complex issue. In 1990, 466 million passengers a year were flying on U.S. airlines. In 2002, the airlines carried 714 million paying passengers. With more airlines and more daily flights, the air traffic control system is dealing with more difficult problems every year. The busiest airports (Atlanta tops the list) cause even more complications—trying to schedule hundreds of flights per hour.

Traffic control is organized into three levels: nationwide U.S. airspace, 20 regional air traffic centers, and individual airports. Air traffic control operators at each airport have immediate control over takeoffs and landings. Regional operators watch traffic within their defined airspace. They “hand off” planes as they fly across the country into the next airspace. Systemwide control is provided by the Central Flow facilities located in Washington, D.C. The Central Flow managers examine traffic across the entire United States and resolve conflicts and problems that arise among regions. The 40 traffic management specialists plan each day in advance, devising alternative routings for aircraft that may be needed because of problems arising from snowstorms, accidents, and closed runways.

Early Systems and Ongoing Problems

The early traffic control system was built with hardware and software from Sperry-Rand/Univac, a computer company that was purchased in the mid-1980s by Burroughs, and now named Unisys. The airport-based traffic control computers were based on 256K bytes of main memory and performed 500,000 instructions per second. The original systems were installed in the early 1960s. The 20 regional centers had their own computers—IBM 9020 machines that were custom made for the FAA in the 1960s.

Air traffic controllers have been reporting problems with existing systems for years:

- In 1992, West Coast air traffic was delayed for several hours. An IBM 3083 at the regional station crashed. In the process, it removed the identification labels from the radar screens of controllers from Oregon to Los Angeles. The controllers switched to an older backup system but had to increase plane separation from the typical 3 miles to up to 20 miles. Pilots and controllers used radio communication and manually filed flight plans to compensate for this loss. Ron Wilson, a spokesman for the San Francisco airport, noted that although there were frequent disruptions, “the FAA computer failures generally don’t last long, just long enough to screw things up.”
- In Oakland, California, the controller screens fail an average of three times a month. When this happens, the controllers have only a few seconds to memorize the position, speed, course, altitude, and destination of the 12 planes they are typically directing. Then their screens go blank for at least 10 seconds. Sometimes when the screens come back online, they are missing critical data.
- Joel Willemsen, assistant director of the U.S. GAO’s Information Management and Technology Division, reported that 70 percent of the 63 largest airports in the United States have experienced problems with blank or flickering computer screens. John Mazor, a spokesman for the Airline Pilots Association, notes the problems cause “delays, diversions, and—in the worst possible cases—accidents. It’s not as dangerous as you might think, but it’s not something you want to have happen to you.”
- The Los Angeles basin region consists of 21 airports handling 6.5 million flights a year. The GAO notes that the FAA computers in the region have repeatedly suffered from the loss of critical data and slow responses because of the overload.
- In 2007, the FAA’s National Airspace Data Interchange Network (NADIN) flight-planning system in Atlanta shut down for several hours, causing flight delays and cancellations across the East Coast. Investigators were unable to find the cause of the problem, but NADIN was scheduled for replacement in 2008 (Weiss 2007).

Improvements

In 1981, the FAA was given approval to upgrade to a comprehensively new computer system. New airports, such as Dallas-Fort Worth, and the deregulation of the airline industry in 1978 led to huge increases in air traffic. The \$12 billion plan called for replacement of 12 major systems over the course of 12 years. An additional 80 smaller projects were included in the plan.

By 1990, only 1 of the 12 systems had been replaced and the project was \$15 billion over the original budget. The one project that was completed was known as Host, because it called for replacement of the mainframe computers at the 20 regional control centers. IBM installed its 3083 mainframes on schedule but was \$16 million over budget. The 3083s were technologically obsolete at the time they were installed because the newer IBM 3090-class machines had already replaced them over a year before.

The FAA has been criticized for a lack of oversight and control in developing new systems. In 1980, the Senate Appropriations committee noted that “the FAA has no ongoing, welldefined, and systematic management approach to evaluating software and operational cost, capacity, and performance of the current system to

meet projected shortrange workloads.” The General Accounting Office (GAO), the watchdog of Congress, echoed that sentiment several times later.

Advanced Automation System

One of the more visible components of the plan to refurbish the system is the Advanced Automation System (AAS). It was designed to provide updated tracking displays for the controllers. It was supposed to be completed by 1990, but at that time was delayed until 1993. The system was designed to use IBM RS-6000 computers to display flight information, schedules, and current location along with weather fronts. The color systems were to have higher resolution, be easier to read, and carry more information.

In 1994, an internal study of the AAS showed that the project was still two years behind schedule and probably would fall back another two years before completion. Up until that time, the project had cost \$2.3 billion. It was estimated to eventually cost about \$7 billion. David Hinson, FAA administrator, announced that he was replacing top managers on the project, dropping portions of uncompleted work, and demanding performance guarantees from the contractors. The Area Control Computer Complex was canceled at this time. It was designed to interconnect the host computers at the airport with those at the regional levels.

Global Positioning System (GPS)

GPS is a satellite-based navigation system that was developed by the Pentagon and previously available for use only in connection with military air travel. GPS allows pilots to navigate based on satellite signals instead of radar signals. It allows real-time flight planning for pilots. As more satellite technology becomes available, the integration of air traffic as well as weather information and other data communication will become a necessary technological step. Four-dimensional GPS readings—longitude, latitude, altitude, and time—enable an aircraft to come within a few feet of any given target. Encryption technology is currently in place to protect security in the transmission of the satellite messages.

In 2004, the FAA began testing GPS tracking for air traffic control in Alaska. Because of the vast rugged terrain, it would be impossible to put radar stations across Alaska. Moreover, the onboard GPS units can report position data every second, while radar hits a plane only once every six seconds. The ADS-B technology GPS systems connect through the Iridium satellite system (Jackson April 26, 2004).

Standard Terminal Automation Replacement System (STARS)

“STARS is the next big step in the FAA’s comprehensive effort to upgrade air traffic control facilities across the nation. The new system will provide the platform for improvements to handle the ever-growing volume of air traffic safely and efficiently well into the 21st century,” said FAA administrator David R. Hinson (Dorr 1996). STARS will standardize all air traffic control equipment at the 172 FAA facilities as well as the 199 Department of Defense facilities. STARS will supply new hardware and software to these facilities. The program will be a complete replacement for the aging systems currently in use.

The most important feature of the STARS system will be the ability to display transmissions. The Automated Radar Terminal System (ARTS) that is currently in place was developed in the 1970s and 1980s. The FAA believes that interim programs are limited in their ability to extend the ARTS life in the short term. It is

generally accepted that this system does not have the capabilities to take air traffic into the next century. ARTS software contains various versions and languages that are very labor intensive as well as expensive to support.

The STARS program includes a commercial standard system that the FAA believes will be much cheaper and easier to maintain. A key feature is the ability to extend and advance the capacity of the system without reengineering the basic architecture. By building on commercially available hardware and software, the development time for the software will be reduced significantly. The resulting maintenance costs will also be lower than those associated with the current ARTS system.

By 2003, the STARS project was behind schedule by at least six years and millions of dollars. The system was supposed to be completed in 1998 for \$12 million. But after more than six years of development the system was still not implemented (McCartney 2003). However, an initial version of the system was installed in Philadelphia in late 2002. The system gathers data from several radar systems on color displays. However, not everyone was happy with it. Controllers in El Paso noted that the system could not distinguish between planes sitting on the runway and trucks on a nearby highway (*CNN* November 17, 2002). In mid-2004, the FAA announced that it was ready to begin implementing the new system. The Phase I rollout would take place at airports with the oldest equipment and cost \$1.4 billion. Nineteen of the fifty sites were online as of 2004. But, the last of the 50 airports were not scheduled to receive the new equipment until the end of 2007. There was no budget or schedule for the remaining 100 plus airports. In 2004, the GAO and inspector general urged the FAA to gain control over costs. The project was already seven years behind schedule and estimated costs had risen to \$1.9 billion (Mosquera April 26, 2004).

Wide Area Augmentation System (WAAS)

The Wide Area Augmentation System is used in conjunction with GPS. Using a network of 36 ground stations to “distill” satellite GPS signals, WAAS will allow commercial aircraft to pinpoint a location within seven meters. With the use of WAAS/GPS, the FAA hopes it can close many of its ground control centers and allow pilots to fly more direct routes. Consolidated, these tools are projected to lead to the concept of free flight.

The WAAS system fell even further behind than the STARS project. The satellite-based system was pushed back by five years and the estimated costs were tripled (McCartney 2003).

Free Flight

Free flight is a consolidated goal toward which the FAA is working. Free flight would enable pilots to control their own navigation procedures. The pilot would use the WAAS and GPS systems for navigational purposes and choose their own routes, speed, and altitude. Ground support will be held to a minimum and would be most important when flights are in congested airport areas, when airplanes approach restricted airspace, or when safety is at stake.

Two principles that drive the free flight plan are the protected and the alert airspace zones. The sizes of these zones are determined based on aircraft speed, performance characteristics, communications, navigation, and surveillance equipment. The protected zone is the zone closest to the aircraft. No aircraft should overlap the protected zone of another aircraft. The alert zone is one that extends

far beyond an aircraft's protected zone. The distance between planes will be monitored closely. If a plane touches another plane's protected zone, the pilots and the air traffic controllers will determine the course corrections that are needed. Under the free flight system, interference will be minimized until the alert zones collide.

Of course, after September 11, the issue of free flight is probably obsolete. The FAA and security agencies are even more interested in controlling and restricting flights. Nonetheless, the FAA and the GAO continue to investigate free flight options. A main step in the process is the Traffic Management Advisor. This software helps controllers efficiently regulate the space between airplanes as they arrive at airports. Under Phase I of the free flight program, five software tools are being tested at various sites. Phase II represents the expansion of the systems—if they work. One system, the User Request Evaluation Tool (URET), was deployed late, so it will require additional testing. It is designed to identify conflicts and respond to pilot requests for route changes. Another tool, the Final Approach Spacing Tool (FAST), has been abandoned because of risks found in testing. It was designed to assign runways and schedule landings (Langlois October 2001)

Some researchers note that reducing flight times will not be sufficient to speed up the system. Delays are also created by slow operations at the terminals, including refueling, baggage handling, and unscheduled gate changes. These researchers suggest that significant changes are needed to improve communications among airport terminals. One possibility is wireless PDAs carried by all personnel and updated by the airlines.

The September 11 attacks caused the FAA to delay implementing some aspects of the free flight (CPDLC) deployment. A major reason for the delay was due to the costs that would be imposed on the airlines. The FAA was also not ready to implement the new technologies (Vasishtha 2002)

Technology Innovations

The FAA has suffered through several failed projects over the years, including the Advanced Automation System (AAS) that was designed in the mid-1990s and thrown away in favor of the STARS project. The FAA also designed and implemented new radio communication technology. The goal was to transfer data by text, to reduce the use of voice communications. The Aircraft Monitory System (ACMS) was designed to collect data on the plane and send it to controllers. The Aircraft Communication Addressing and Reporting System (ACARS) was introduced to cut down on the use of spoken radio messages to transmit information to the ground. It was thought that if the flight crew could save time by transmitting data to the ground rather than conveying it by voice to the air traffic controllers, they would be better able to concentrate on flying the plane. ACARS directly interfaces with ACMS and sends and receives messages directly to and from the pilot. The pilot punches the message, such as flight plans, in an alphanumeric keypad or touch screen. Both systems operate on the Aeronautical Radio system (ARINC) that runs on VHF radio waves and handles the data transmission between the plane and the ground controllers. The system is owned and operated by the major airlines. The main drawback to ARINC is that because of limited bandwidth, the system transmits data at 2.4 kbps. In 2004, some airlines (notably Southwest) began installing a newer data service called VHF Digital Link Mode 2 (VDLM2), which can transmit data at rates up to 31.5 kbps (Brewin 2004).

Launched in 2003, the FAA budget for 2008 contained \$175 million earmarked for the Next Generation Air Transportation System (NGATS). The system relies

more on satellite (GPS) navigation. It also encourages airports to upgrade their facilities

Network

In 1998, the FAA replaced its mainframe-based system for acquisition management with a distributed architecture. The old system ran on 1980s-era minicomputers at 12 centers nationwide and processed more than 200,000 purchases per year. It was not updated for more than three years and was not Year 2000 ready. Mounting problems in the old system led many FAA officials to revert to paper to track agency purchases.

The new system is called Acquire. It uses Oracle Corporation's Alert software and the Discoverer/2000 querying tool. The FAA must also use Oracle Federal Purchasing software to get Acquire to run on a network that links headquarters to regional offices and field centers.

The FAA also began preparing a communications system overhaul aimed at readying the agency's infrastructure to meet the needs of the 21st century. The FAA Integrated Communications Systems for the 21st century (FICS-21) program is projected to cost an estimated \$2.75 billion.

FICS-21 will provide ground-to-ground transmission switching and network management control for voice, data, and video communications. The new initiative will replace at least 11 major programs, including FAA-owned and leased networks. FAA FICS-21 program manager Jeff Yarnell says it is a good time to rebuild the FAA's telecommunications infrastructure because many telecommunications contracts expired at the turn of the century.

In 2004, the FAA finally began rolling out its new communication backbone. The new FTI system was installed at 27 facilities. Steve Dash, FAA telecom manager, said that the system is replacing five disparate networks. He noted that "it's the first phase. The backbone will tie together the major operation facilities" (Jackson January 26, 2004). Ultimately, the system will be connected to the other 5,000 FAA facilities and save \$700 million in telecommunication costs over 15 years. Installation of the system was contracted to Harris at an estimated total cost of \$3.5 billion. As much as possible, the system will use off-the-shelf networking and telecommunication products. The new FTI system fell more than a year behind schedule and in 2006 and 2007 the agency ended up paying for both systems simultaneously because they had to maintain the old one while transitioning to the new, incomplete system (Sternstein 2006).

The FAA also provides services to pilots (and the public) through its Web site. Pilots account for 30 percent of the site traffic. To provide faster service, the FAA installed an expert system from RightNow Technologies that examines questions posed by visitors. The software compares the question to answers that have been provided to other users. Matches that are close are immediately displayed to visitors. Other questions are forwarded to the appropriate FAA authorities. Greg Gianforte, CEO with RightNow comments that "we use a series of both implicit and explicit learning capabilities, which include artificial intelligence and machine learning, to observe the historical usefulness of each knowledge item and provide greater visibility to knowledge." Typically, the system can automatically handle 90 percent of the inquiries (Chourey April 26, 2004).

In conjunction with NASA, the FAA is using a simulation system called FutureFlight to test changes to airport control systems. Researchers testing configurations of the LAX airport found that safety could be improved by moving a

taxiway to one end of the airport. John Bluck, speaking for the Ames Research Center, notes that “the idea is to try it [changes] in a safe way that’s as close to reality as we can make it. You don’t have to try something new on a real airport, where you have thousands of flights coming and going” (Langlois October 2001).

The Future

The FAA has faced considerable criticism over the delays and cost overruns associated with replacing its primary systems. The agency makes heavy use of outside contractors, which is probably a necessity. However, the agency needs to write better contracts so that it can maintain control over costs and schedules.

The successful implementation of STARS is becoming critical. Like other federal agencies, by 2014, as many as half of the air traffic controllers can retire (about 7,000 people) (Chourey July 5, 2004). These workers require intensive training, and their salaries represent a significant expense. In 2002, more than 1,000 controllers earned over \$150,000 (McCartney 2003). The FAA is going to need better automated systems that are easier and safer. With increased traffic demands, the FAA will have to find a way to improve productivity.

In the 2008 budget year, the FAA is pushing for a new funding mechanism. In part led by the commercial airlines who are afraid of the microjet market, the FAA is trying to push for a cost-based user fee system. Direct fees to commercial carriers and fuel excise taxes collected from general aviation would be determined by the FAA using some unspecified process to match the fee to costs of the services provided to the two user groups. The FAA also wants the authority to levy additional fees for the most congested airports (2008 U.S. Budget). The agency claims that the NextGen project cannot be built without more funding. Planned for completion in 2025, the project is estimated to cost \$15 to \$22 billion (Bain 2007). In 2007, the FAA awarded an initial design contract to ConceptSolutions, LLC for a five-year \$32 million project to design the NetGen system (Hardy 2007). The FAA claims the system is vital to increase flexibility and handle the anticipated 30 percent increase in flights. The 2012 proposed Federal budget called for \$1.24 billion funding for NextGen, an increase of \$370 million compared to 2010.

In 2000, Congress approved creating an internal manager to oversee the flight-control operations, but the position went empty for three years until the FAA hired Russell Chew in 2003. Coming from business, Mr. Chew has removed layers of bureaucracy, instituted cost measurement programs, and attempted to instill business management into the system. He ordered the first inventory of equipment ever conducted at the FAA. With measurements in place, the FAA determined that it cost \$457 to handle one jet on one flight in 2003. With cost-cutting measures, he reduced the number to \$440 in 2004. He has also tried to reduce costs, by urging for consolidation of facilities. But, Congress ultimately controls spending and representatives tend to fight plans that call for a reduction of jobs in their districts. Mr. Chew also faces resistance from other FAA managers. Marion Blakey, FAA Administrator observed that many FAA employees “see any kind of change as very threatening” (Meckler 2006).

The 2010 discretionary spending for the FAA was \$12.478 billion. The fiscal year 2012 proposed budget calls for a slight increase to \$12.883 billion. These numbers are on top of the mandatory spending of \$3.515 billion in 2010 and proposed \$2.424 billion in fiscal year 2012. The big challenge is that the FAA does not plan to fully replace the existing radar tracking system until at least 2025. In the meantime, problems still arise, such as the minor crash in 2011 when an Air

France Airbus A380 (a huge commercial airliner) collided with a regional CRJ-700 jet while taxiing at the JFK airport in New York. Bill Voss, former FAA air traffic development director and current president of the Flight Safety Foundation noted that “You’d be surprised—almost all of this is done with pieces of paper, an eyeball and a pencil. It is a very visual and manual activity.” There were also a string of incidents (but not crashes) in 2011 when late-night traffic controllers were reported to be sleeping on the job (Patterson 2011). In terms of good news, most ocean flights controlled by the U.S., Canada, New Zealand, Australia, and most of Europe, are already monitored and controlled using a GPS system. The Advanced Technology and Oceanic Procedures (ATOP) system was adapted by Lockheed Martin from a system developed in New Zealand. After four years of use, with pilots and airlines defining their own routes, the system has saved nearly 10 million gallons of fuel. (McCartney 2009).

Questions

1. After 20 years, why is the FAA still having so many problems building new systems?
2. What tools or methodologies might be useful to help the FAA complete its remaining tasks?
3. Is it possible to speed up the NextGen schedule?
4. Will outside contractors (Computer Science Corporation) help the projects? How can you monitor and control the work by the contractors?

Additional Reading

“Air Traffic Control - Good Progress on Interim Replacement for Outage-Plagued System, But Risks Can Be Further Reduced,” *GAO Report*, October 17, 1996.

Bain, Ben, “FAA Funding Battle Threatens to Ground NextGen,” *FCW*, June 14, 2007.

Brewin, Bob, “Data Takes Flight,” *Computerworld*, June 21, 2004.

Chourey, Sarita, “Getting an Instant Response,” *FCW*, April 26, 2004.

Chourey, Sarita, “Air Traffic Controllers Head To the Beach,” *FCW*, July 5, 2004.

CNN Online, “New Air Traffic Control System Tested,” November 17, 2002.

Dorr, Les. “FAA Selects Raytheon for Next-Generation Air Traffic Control System Upgrade,” *FAA News*, September 16, 1996.

“FAA Radar Glitches Found.” *Computerworld*, November 2, 1998, p. 12.

“FAA Ready for Free Flight,” *Advanced Transportation Technology News*, April 1996.

“FAA’s \$500 Million Navigation Contract Takes Flight,” *Federal Computer Week*, April 10, 1995.

Hardy, Michael, “FAA Taps ConceptSolutions for Next Generation Air System,” *Government Computer News*, April 6, 2007.

Jackson, Joab, “FAA Considers Commercial Satellites for Traffic Control,” *Government Computer News*, April 26, 2004.

- Jackson, William, "FAA and GSA renew their dogfight over air traffic control modernization," *Government Computer News*, September 18, 1995, p.73.
- Jackson, William, "FAA Clears Rollout of Net Backbone," *Government Computer News*, January 26, 2004.
- Langlois, Greg, "Researchers: Don't Forget Airport Ops," *Federal Computer Week*, September 10, 2001.
- Langlois, Greg, "Data Determining Free Flight Future," *Federal Computer Week*, October 2, 2001.
- Langlois, Greg, "NASA: Simulation Proves Its Worth," *Federal Computer Week*, November 5, 2001.
- Leopold, George, "Study: GPS can Fly as Commercial Air Navigator," *Electronic Engineering Times*, February 8, 1999, p. 18.
- McCartney, Scott, "Labor Costs, Mismanagement: Airlines Again? No, the FAA," *The Wall Street Journal*, February 19, 2003.
- McCartney, Scott, "Building a Better Air-Traffic-Control System," *The Wall Street Journal*, October 22, 2009.
- Meckler, Laura, "FAA's Mr. Chew Tackles High Costs, Chronic Lateness," *The Wall Street Journal*, January 30, 2006.
- Mosquera, Mary, "FAA Slows Down Display System Deployment," *Government Computer News*, April 26, 2004.
- Patterson, Thom, "Can Technology Fix the FAA's Air Traffic Troubles?" CNN Online, April 26, 2011.
- Sternstein, Aliya, "FTI Delay Leaves FAA Paying for Two Systems at Once," *FCW*, December 8, 2006.
- Tiboni, Frank, "FAA Begins Upgrade Project On Its Controller-Pilot Comm System," *Government Computer News*, February 8, 1999, p. 8.
- Vasishtha, Preeti, "FAA Defers CPDLC Deployment," *Government Computer News*, January 4, 2002.
- Weiss, Todd R. "FAA Still Seeking Cause of Flight Plan System Shutdown," *Computerworld*, June 12, 2007.

Case: The Internal Revenue Service (IRS)

For the 2009 tax year, the IRS processed more than 144 million individual and 2.4 million corporate tax returns (www.irs.gov/taxstats). Many of the returns are simple one-page forms; others run to thousands of pages of supporting documents. Overall, the service handles more than 1 billion information documents a year. The IRS processes more than \$1 trillion in tax revenue a year. The IRS has 10 regional service centers that are responsible for processing and storing individual forms.

Until 1990, all documents at the IRS were stored as paper records in a central warehouse. In 1989, it cost the IRS \$34 million just to store the paper documents. Documents were organized according to the year of filing. As a result, if

a taxpayer had a problem or question that covered multiple years, the citizen had to schedule multiple meetings with IRS officials to correct problems for each of the years. In some cases, it could take weeks or months just to get the files. Occasionally, the IRS found it was faster to ask the taxpayer for a copy of the return. By the early 1990s, this problem was resolved by having each of the 10 service centers store digital images of the tax returns, making them available to agents on their terminals. While a step in the right direction, this approach did not give the IRS the flexibility it would receive from the ability to scan the returns directly into a computerized information system.

Automation sometimes causes problems in addition to solving them. Such was the case of Dickie Ann Conn. The IRS determined that she owed \$67,714 in back taxes. She was sent a bill for more than \$1 billion in interest and penalties. After being challenged, the IRS admitted that there was an error in the interest computation.

The IRS operating budget in 2010 was \$12.146 billion, with a proposed fiscal year 2012 budget of \$13.284 billion.

A History of Automation Problems

The IRS seems like a logical candidate for improved automation. The benefits of faster processing, fewer mistakes, and easier access to data ought to save a considerable amount of money. The computer's ability to search the data, automatically match transactions, and analyze each return presents several additional opportunities that can either cut costs or raise additional revenue. Managers at the IRS are fully aware of the potential, and they have proposed several systems over the years. The problem has been in implementation of the plans and in getting Congress to financially support the changes.

In the late 1960s, the IRS knew it needed to redesign its basic systems. In response, it began to plan for a system to be installed in the 1970s. The IRS did not get the needed support in Congress because of fears that it would be too expensive and too invasive into individual security and taxpayer privacy. As a result of this lack of support, the IRS turned its attention toward keeping its existing computers running.

In 1982, the existing system was nearing capacity and the IRS established the Tax System Redesign program. It promised a complete redesign of the system. According to the GAO, changes in management resulted in the system never getting past the design stage. A new assistant commissioner in 1982 embarked on the design of a new system that promised to carry the IRS through the 1990s. Initial costs were estimated at \$3 billion to \$5 billion over the entire project. The primary objective was to replace the old central tape-based system with an online database. Eventually, optical technology would be used to scan the original documents and store the data in the database. A new communication system would carry the data to any agent's workstation. By 1989, initial planning had already cost the IRS more than \$70 million, with no concrete proposal or results.

The main computer systems were replaced at the IRS service centers in 1985. The change in the systems was almost disastrous for the IRS. It delayed returns processing and led to delays in refunds that cost the IRS millions of dollars in interest payments. IRS employees worked overtime but still could not keep up. Rumors were flying that some employees were dumping returns to cut down their backlog. Because of the delays and backlogs, the IRS managed to audit only about half the usual number of returns on which it conducted audits.

In 1986, the IRS initiated a plan to provide 18,000 laptop computers to enable its field auditors to be more productive with its Automated Examination System (AES). Unfortunately, the service bought the Zenith laptops a full year before the software was ready. The system was written in Pascal and was delivered to agents in July 1986. It was designed to examine Form 1040 returns. The biggest drawback was that it used 18 different diskettes. This required agents to be constantly swapping disks. Based on the privatization directives from the Reagan administration, the system was subcontracted to outside developers. As IRS funding was cut, programmers with experience in Pascal were cut. This led the system to be rewritten in C.

A survey in 1988 revealed that 77 percent of the agents were dissatisfied with the software. Only 33 percent said that they used it. By 1989, the IRS revised the software and managed to reduce it to eight disks. By this time, the AES project was more than six years behind schedule and, according to the GAO, was \$800 million over the original budget. The IRS originally anticipated that the AES would produce \$16.2 billion in additional revenue over nine years by making agents more productive. The GAO disputed those numbers, noting that “the IRS has been unable to verify that the use of laptops has actually resulted in the examination of additional returns or increased tax revenues.” In 1990, the White House cut the funding for the program from \$110 million to \$20 million.

In 1999, the IRS implemented a new network to connect computers throughout the organization. Twenty staffers were dedicated to the project and took four years to complete it. IBM’s Tivoli software is a key tool to manage the 132,000 networked devices in 87 locations. The software enables network managers to continually monitor all aspects of the network. They can also push changes down to the desktop computers if problems arise or they need upgrades. Before the system was available, it took an IRS staff member 20 minutes to update each device. With Tivoli live in 2003, a single network administrator sent one update to 400 desktops in one minute. Jim Kennedy, program manager for enterprise systems management at the Austin, Texas, support center estimates that the system has saved \$2.6 million in the first quarter alone (Dubie 2003).

Technology Innovations

By 1989, the IRS knew that it desperately needed to redesign its entire system for collecting taxes and processing information. In hearings before Congress, Senator David Pryor (D-Ark.) noted that the 1960s-era IRS computers were headed for a “train wreck” in the mid-1990s. The GAO estimated the total project would cost between \$3 billion and \$4 billion. The projected date for implementation slipped from 1995 to 1998.

The overall design for the Tax System Modernization program (TSM) called for a centralized online database, smaller departmental systems containing local information, and linkage through a nationwide network. Tax return data would be entered through a combination of electronic filing and optical scanners.

By 1991, the estimated cost of the plan had expanded to \$8 billion. Although the IRS projected that the system would cut \$6 billion in costs, the plan was rapidly attacked by members of Congress. Three studies of the TSM plan by the GAO were released in early 1991:

- The GAO was concerned that optical technology was not sufficiently advanced to perform the tasks demanded by the IRS. The GAO urged greater emphasis on electronic filing.

- The GAO was concerned that management issues such as transition planning, progress measurement, and accountability were not sufficiently addressed by the plan.
- The GAO and Senator John Glenn (D-Ohio) voiced concerns about data security and integrity.

GAO official Howard Rhile noted, “This is a serious omission in view of the fact that the IRS intends to allow public access... to some of its systems and because concerns over the security of taxpayer information helped doom the first [IRS] modernization effort in the late 1970s.”

Despite these misgivings, the IRS was committed to the TSM plan. Fred Goldberg, IRS commissioner, agreed with the GAO findings but observed that

We have been running our business essentially the same way, using essentially the same computer and telecommunications systems design for 25 years. [Existing systems] will perform well and achieve incremental improvements for the next few years... Our best judgment is that [OCR] technology will be there when we need it, by the end of the decade.

By 1992, the situation grew worse. Shirley Peterson, the new commissioner of Internal Revenue, stated at a congressional hearing that

Our systems are so antiquated that we cannot adequately serve the public. The potential for breakdown during the filing season greatly exceeds acceptable business risk.... Some components of these computers are so old and brittle that they literally crumble when removed for maintenance.

In December 1991, the IRS awarded a 12-year, \$300 million contract to TRW to help manage the process and provide planning and system integration services. The recommended system was ambitious. It called for 60 major projects, two dozen major purchases, 20 million lines of new software, and 308 people just to manage the purchasing process. Despite the best efforts of the administrators, elements of the IRS modernization plan were stalled because of purchasing difficulties. In July 1991, the IRS awarded a billion-dollar Treasury Multiuser Acquisition Contract (TMAC) to AT&T. The goal was to standardize purchasing for the IRS and the Treasury Department by routing all purchases through one vendor. The contract was challenged by other vendors and overturned. The contract was rebid and AT&T won the second time. IBM (one of the original protesters) again objected to the process, noting that the IBM bid of \$708 million was less than the \$1.4 billion bid by AT&T.

In 1993, the IRS acknowledged that the TSM Design Master Plan needed to be rewritten. In particular, it had to focus on business aspects instead of technology elements. To better coordinate technical planning with IRS needs, the agency established a research and development center funded by \$78.5 million of federal money but run by the private sector. The center was responsible for providing technical assistance and strategic planning for the TSM. The IRS also established a high-level “architect office” to evaluate technologies and direct their proposed uses.

Throughout calendar year 1992, the IRS spent \$800 million on TSM. In 1993, new IRS estimates indicated that TSM would cost \$7.8 billion above the \$15.5 billion needed to keep existing systems running. The new system was projected to generate \$12.6 billion in total benefits by 2008 through reduced costs, increased

collections, and interest savings. Moreover, the improved process was supposed to save taxpayers \$5.4 billion and cut 1 billion hours from the collective time they needed to spend with the IRS.

In 1996, the IRS asked Congress for a \$1.03 billion appropriation. This was a substantial increase over the \$622 million it spent on automation in 1995. Hazel Edwards from the General Accounting Office noted, “After eight years and an investment of almost \$2 billion, the IRS’s progress toward its vision has been minimal.”

IRS Commissioner Margaret Milner Richardson denied the GAO claims. She noted, “I think we have made significant progress, not minimal progress... but we do know we can and must do more” (Birnbaum, 1998).

The IRS situation represented a dilemma for Congress. The IRS claims that the only way to make a system that works is to spend more money. The GAO has set forth that it is impossible to complete the entire project envisioned by the IRS. The GAO believes the IRS should, instead, concentrate on smaller, more focused projects that can be completed in a one- to two-year timeframe.

In 2001, Congress passed tax-cut legislation to stimulate the economy, and ordered the IRS to send “refund” checks to all taxpayers. It took several months to create and mail the tens of millions of checks, but most of them were correct. On the other hand, about 523,000 taxpayers received notices that they would be getting a check for the full refund amount, when they were actually eligible for only part of the refund. The mistake was attributed to a programmer error, and the final checks were correct; but some taxpayers were confused by the misleading letter.

Electronic Filing

The IRS introduced electronic filing in 1986, when 25,000 forms were filed electronically. By 1990, 4.2 million people filed for tax refunds electronically. In 1992, the number increased to 10 million filers. In 2003, 49 percent of the personal tax returns were filed electronically (www.irs.gov).

The primary target for electronic filing is the millions of individual taxpayers who are slated to receive refunds. To control the process and ensure that documents are properly filed, electronic filing is available only through authorized tax preparers. The IRS is deliberately avoiding providing access to individual taxpayers. As a result, taxpayers who use the system pay an additional charge to the preparer. However, the electronic filing system provides refunds within a few days.

Forms that have been electronically filed cost the IRS one-tenth the processing cost of paper forms. This approach also eliminates the cost of paper storage. The IRS notes that it is able to store 800,000 returns on one side of a 12-inch optical disk.

For taxpayers with easy returns, the IRS is simplifying the process even further. Short forms can now be filed over the telephone. In a 1992 pilot, 117,000 Ohio taxpayers filed for refunds using push-button phone calls. The system was expanded nationwide in 1994. The push-button system can be used only by taxpayers who are able to use the 1040EZ form. A replacement form (1040-TEL) must still be signed and filed with the IRS, along with the W-2 (withholding) statements.

In the 1998 IRS Restructuring and Reform Act, Congress required the IRS to encourage the use of electronic filing. The IRS has made it easier for people to file electronically—particularly for those who use computer software to compute their taxes. In 1998, about 20 percent of individuals filed electronically; in 2000

the number was 28 percent; in 2001 about 32 percent (45 million). The IRS goal is to increase this number to 80 percent by 2007 (Dorobek 2001). For the 2001 tax year (filing in early 2002), the IRS used the Digital Signature law to send PINs to several million taxpayers, enabling them to legally sign their tax forms electronically. However, the one important catch is that taxpayers who file electronically must pay an additional fee to do so. Hence, only those who receive refunds (about 70 percent of the filers) are interested in paying the fee, because it enables them to get their money faster. Most experts believe it is unlikely that the IRS will meet the congressional goal of 80 percent by 2007.

In 2009, the IRS noted that it cost \$3.29 to process a paper return and only 19 cents to process one filed electronically (Saunders March 2011). In 2011, the IRS stopped mailing paper forms to people to encourage them to file electronically. For 2009, about 70 percent of the forms were filed electronically. But e-filing is least used by wealthy taxpayers, and they are the ones with the most complex returns. Some people with complex forms cannot file electronically. Other people are wary of turning over massive amounts of data to a third-party company to handle their taxes online and submit them to the IRS.

The Internet

In late 2001, the IRS announced plans to offer electronic payments by businesses over the Internet. A major portion of the money received by the IRS comes from withholdings collected by businesses. This money has to be forwarded to the IRS at regular intervals, so the IRS is trying to reduce handling costs by moving these transactions online. The Electronic Federal Tax Payment System (EFTPS) is a Web-based system that can also be used by small businesses and by taxpayers who make estimated quarterly payments. Using modern strong encryption technologies, the IRS is confident the system will be secure.

Relatively early in the dot-com and dot-gov restructuring, the IRS realized the importance of putting information on its Web site. In fact, a huge amount of information is available online. And that is a problem. In 2001, IRS executives were asked to search the site for common tax information. It generally took 20 or 30 clicks to find any piece of information. To improve its Web site, the IRS hired Gregory Carson in 2001, a designer from private industry who helped to launch the Priceline.com Web site.

The IRS also signed a contract with the consulting group Accenture to redesign the IRS Web sites. In 2001, the site received 80 million hits a day. Gregory Carson, director of electronic tax administration modernization at the IRS, notes that “the development of an intuitive, intentions-based design will make it considerably easier for taxpayers and tax preparers, who pull forms from the site, to obtain the information and documents they need to file tax returns.” (Rosencrance August 2001) Accenture’s goal is to make the site easier to use so that users can reach the desired information within three clicks. Furthermore, Accenture will be hosting the site on its servers.

In 2003, more people turned to the Internet to file their tax returns. Several companies provide online systems that automatically e-file the data with the IRS. A few offer free filing. In 2003, 3.4 million taxpayers used the Free File service. In total over 14 million people used their personal computers to e-file their taxes (Mosquera May 10, 2004).

Automated Under-Reporter (AUR)

The Automated Under-Reporter (AUR) is another component of the TSM. The AUR is a system designed to monitor returns and identify people who are most likely to underpay their taxes. The system was first installed in 1992 at the Ogden, Utah, regional center. The system pulls data from the service center's Unisys 1180 mainframe. The data is downloaded across a local area network to a Sequent Computer System S-81 minicomputer. From there the information is sent to one of 240 networked UNIX workstations on the employees' desks.

The system automatically matches distribution documents (such as 1099s and W-2s) with the filings of individual taxpayers. Mark Cox, assistant IRS commissioner for information systems development, noted that in trials with the AUR, "we've been able to cut down the rework of cases from 25 percent to less than 5 percent. We see this type of work enabling us to share in more of a connectivity mode" (Quindlen, 1991).

The system uses an Oracle database running SQL to match data from various sources. It also performs basic tax computation and helps agents send notices to taxpayers. Managers have noted that even though the new system has not improved the speed of the agents, it has cut down on the error rates. As agents become familiar with the system, productivity is expected to improve.

In 1991, the Ogden center processed 26 million tax returns and collected \$100 billion in tax payments. It processed \$9 billion in refunds. In 1992, it won the Presidential Award for Quality for improved tax processing by saving the government \$11 million over five years.

In 2007, the IRS estimated that the compliance rate for individuals paying taxes on time was 86 percent—leading to an estimated \$290 billion per year tax gap (based on 2001 data). President Bush argued (2008 U.S. Budget) that if this tax gap could be reduced, many programs could be funded without additional taxes. His budget for the IRS called for an additional \$410 million for research, enforcement, technology, and taxpayer services to reduce this gap.

The Currency and Banking Retrieval System

In 1988, Congress passed a new law in an attempt to cut down on crime (notably drug dealing) and to provide leads to people who significantly underreport their income. Every cash transaction over \$10,000 is required by federal law to be reported to the IRS on a Form 8300. The IRS created the Currency and Banking Retrieval System to match these forms against the filer's tax return. The system automatically identifies people who had large cash purchases but claimed little income. Because of a programming error, the system missed forms covering \$15 million in cash transactions between 1989 and 1990.

The problem stemmed from the fact that the IRS used the same code number on the 8300 forms that it used on other cash transaction forms. The IRS later assigned separate codes for each form. When programmers wrote the new matching programs, they did not realize there were two codes for each transaction. The system was corrected in 1991. By 1992 it was used to process more than 1 million queries a year.

Jennie Stathis of the GAO noted there were additional problems with Form 8300. In particular, the filings were incomplete or contained incorrect taxpayer identification numbers. The IRS is developing software to enable businesses to verify the taxpayer ID numbers automatically before the customer completes the purchase.

Document Processing System (DPS) and Service Center Recognition/Image Processing System (SCRIPS)

In 1994, the IRS awarded a \$1.3 billion contract to the IBM Federal Systems division to design a document processing system. The goal was that by the late 1990s, the system would convert virtually every tax return to a digital format. A day after the contract was awarded, IBM sold the Federal Systems division to Lor Corporation for \$1.52 billion.

The 15-year systems integration contract was to have the system running on-line in 1996. The plan called for scanning incoming tax forms. Special software digitally removed the form layout and instructions, leaving just the taxpayer data. OCR software was to then convert the characters (including handwritten numbers) into computer data.

The system was scheduled for initial installation at the Austin, Texas, regional center in August 1995. Plans called for installing it at Ogden, Utah; Cincinnati, Ohio; Memphis, Tennessee; and Kansas City, Missouri, by 1998. Despite the popularity of electronic filing, the IRS still sees a need for the OCR system. The IRS received 222 million returns in 2003. Of those, 53 million were electronic.

SCRIPS was the first scanning project. Presented at a cost of \$17 million, it was approved to cost \$88 million when it was awarded in 1993 to Grumman Corporation's Data Systems unit. SCRIPS was designed to capture data from four simple IRS forms that are single-sided. SCRIPS was supposed to be an interim solution that would support the IRS until the Document Processing System (DPS) could be fully deployed. However, delays pushed back the delivery of the SCRIPS project. By the time it was declared finished, the project cost \$200 million (Birnbau 1998).

DPS was the second scanning project. It has a projected cost of \$1.3 billion. Interestingly, Grumman Data Systems was the loser in the contest for the DPS contract. The IRS noted that Grumman failed a key technical test. When completed, DPS was quite complicated to use. In this program, the IRS developed nine separate databases, most of which could not communicate with each other.

In 1996, Art Gross, a veteran of the New York State revenue department, became the new IRS chief information officer. He stated that the IRS's computers didn't "work in the real world" and that its employees lacked the "intellectual capital" to transform them. When he arrived in 1996, the IRS's Year 2000 conversion project had a budget of \$20 million and a staff of three; by 1998, it had grown to a \$900 million project with 600 workers, many of them consultants (Birnbau 1998).

Gross tried to get control of the system. He ended the DPS or "Bubble Machine" project as being over budget and behind schedule. With help from TRW, he devised a new top-to-bottom computer architecture. The architecture was built around a centralized database to coordinate information at the IRS.

When Charles Rossotti arrived as the new commissioner, he proposed an even more ambitious plan. In addition to Year 2000 changes, computer updates from the 1997 tax law, and the overall modernization, Rossotti proposed to restructure the entire organization. This proved to be too much for Gross, who resigned.

In 1998, Congress passed the Government Paperwork Elimination Act, part of which forces the IRS to move to more electronic transactions. Since then, the IRS has created electronic versions of its forms that can be downloaded from its servers. In 2001, the IRS signed a contract with ScanSoft Inc. for OmniPage Pro 11 for use in its federal tax offices around the nation. The goal is to convert the mass-

es of paper files into electronic documents. Instead of taxpayer files, the system is designed more to convert internal forms and documents so that all employees will have immediate access to up-to-date forms and policies on the IRS intranet.

Customer Relationship Management

In late 2001, the IRS began installing customer relationship management (CRM) software that it purchased from PeopleSoft. A key element of the kinder, gentler approach is the ability to track customer issues. CRM software can collect all of the customer interactions into one location—making it easier for multiple agents to see the entire history of a particular problem. The system will also enable the agency to create Web portals for professional tax preparers, IRS employees, and taxpayers. The portals will securely provide individual information to these groups over the Web. In addition to faster service, the IRS hopes to reduce the costs of its call centers by moving more access online.

The IRS also developed the e-help system in 2002 to provide a central point for customer service. The system was designed to ensure service representatives provide consistent and accurate responses to customer questions. The Inspector General in 2007 reported that the system had made progress, but still needed improvement. Notably, the system lacked quality measures and procedures, including a failure to survey customer opinions and train employees. In a random sample of 19 employees, the Inspector found that none of them had completed required training. Michael Phillips, IRS deputy inspector general for audit noted that “ensuring assistors complete required training will be of greater important as the IRS moves forward with implementation of the next available technology” (Cranmer 2007).

Security Breaches

In 1983, Senator John Glenn (D-Ohio) released an IRS report indicating that 386 employees took advantage of “ineffective security controls” and looked through tax records of friends, neighbors, relatives, and celebrities at the Atlanta regional IRS office. Furthermore, five employees used the system to create fraudulent returns, triggering more than 200 false tax refunds. Additional investigations turned up more than 100 other IRS employees nationwide with unauthorized access to records. Glenn observed that the IRS investigation examined only one region and looked at only one of 56 methods that have been identified to compromise security. Glenn expressed the concern that “this is just the tip of a very large iceberg.”

The IRS noted that the TSM program “greatly increases the risk of employee browsing, disclosure, and fraud,” because of the online access to the centralized databases.

Margaret Richardson, commissioner of the Internal Revenue Service, noted that the system used by the perpetrators was 20 years old and was used by 56,000 employees. It met all federal security standards, including the use of passwords and limited access based on job descriptions. The IRS found the problems in Atlanta by examining records of database access from 1990 to 1993. Because the system generates 100 million transactions a month, the data is stored on magnetic tape, making it difficult to search.

In 1989, the IRS arrested Alan N. Scott, of West Roxbury, Massachusetts, for allegedly submitting 45 fraudulent returns via the new electronic filing system. The IRS claims Scott received more than \$325,000 in refunds.

The IRS requires tax return preparers to fill out an application before it issues an access code. Scott apparently used a fake taxpayer ID number and lied on the application form to gain the access number. The IRS claims he then submitted false returns using bogus names and taxpayer ID numbers to get refund checks ranging from \$3,000 to \$23,000.

IRS officials noted that the electronic filings actually made it easier to identify the problem, because the computer could scan the data earlier than the data would have been scanned if it had been submitted by hand. Once the situation was identified, the IRS was able to immediately lock out further transactions from Mr. Scott's access number.

In May 2007, the Treasury Department's Inspector General reported that the IRS lost 490 computers between 2003 and 2006. Of these, 111 occurred within IRS offices. Most of the machines lacked encryption and strong passwords. The IRS has 100,000 employees and has issued 47,000 laptops. It was unable to identify the data that was lost because the agency has no records of what data was stored on the machines, but the audit report claims data was compromised for at least 2,300 taxpayers. Deputy inspector general Michael R. Phillips, stated that "we believe it is very likely a large number of the lost or stolen IRS computers contained similar unencrypted data. Employees did not follow encryption procedures because they were either unaware of security requirements, did so for their own convenience, or did not know their own personal data were considered sensitive. We also found other computer devices, such as flash drives, CDs, and DVDs, on which sensitive data were not always encrypted." An audit in 2003 reported similar problems, and the IRS has taken no action to change procedures (Gaudin 2007).

Modern Disasters

In 1998, the message in congressional hearings was to "Do something. Anything." The hearings into IRS dealings with the public revealed several problems within the IRS. They emphasized the negative perceptions the public has toward this important agency. After listening to these criticisms, the IRS eventually agreed to change some of its policies to improve its treatment of citizens. The 1998 IRS Restructuring and Reform Act was aimed at changing IRS attitudes and providing citizens with more control in the tax-collection process. Charles Rossotti, the new IRS commissioner, described the process of upgrading the vacuum tube-era technology as being similar to "rebuilding Manhattan while we're still living in it." The \$7 billion agency has attempted the same gargantuan task of modernizing its computers for 25 years and continues to fail. The total cost in the 1990s alone has been projected to be nearly \$4 billion (Birnbaum 1998).

In 2002, the system included 80 mainframes, 1,335 minicomputers, and 130,000 desktop boxes that were largely unable to communicate with each other. Before his appointment as commissioner of the IRS in November 1997, Rossotti served as chairman of the computer consulting firm American Management Systems. In early 1998, Arthur Gross, the chief technology officer, who drafted the latest modernization blueprint, resigned in frustration. Shortly thereafter, Tony Musick, the chief financial officer, resigned to become deputy CFO at the Commerce Department (Birnbaum 1998).

Unfortunately, the IRS has been even less successful at implementing new technologies. By 1998, nearly all of the earlier systems development efforts were canceled. In late 1998, the IRS signed a 15-year development contract with Com-

puter Science Corporation (CSC) that was worth \$5 billion. By contract, CSC is responsible for helping design new systems, indicating that the ultimate goal is still to be determined. Outside experts note that the contract does not necessarily solve all the IRS problems. The IRS must still deal with the contract management issues, which have proved difficult to the IRS in the past.

In 1999, the IRS launched yet another attempt to modernize its systems. The \$8 billion Business Systems Modernization (Bizmo) program was supposed to replace the infrastructure and over 100 applications. A key element is to replace the Master File system—which is an ancient tape-based system that holds customer data that the IRS has been using for over 40 years. The system runs an archaic programming language with code written in 1962. The heart of the new system is the Customer Account Data Engine (CADE) designed to run on IBM's DB2 database system. As of 2004, the project is way over budget and years behind schedule. Even the system to process the simple 1040EZ form is three years late and \$36.8 million over budget (Varon 2004).

The system design actually started out well. The IRS hired CSC as the prime contractor. But the IRS did not maintain control of the contract, and there are serious doubts that even CSC was capable of handling the complex project. Paul Cofoni, president of CSC's Federal Sector business, testified to the U.S. House Ways and Means Oversight Subcommittee that "I have never encountered a program of the size and complexity as the Business Systems Modernization program at the IRS" (Varon 2004). Several times, the IRS considered firing CSC, but kept deciding that it would not be cost effective. One of the problems is that the IRS is not providing sufficient oversight of CSC or the project. They originally planned a relatively hands-off approach to let the company use best practices in its development. The problem is that CSC needed the expertise of the IRS agents and IT workers. The other problem is that the IRS went through five CIOs in seven years. In the meantime, CSC has gone through four managers to lead the project.

The CADE system is an impressive piece of technology—if it ever works. Once the database is active, the IRS will use a customized version of the Sapiens eMerge rule-processing engine. Congressional tax laws are coded as business rules that the system applies to evaluate each tax return. The system includes an XML-based RulesScribe layer that handles changes and additions to the rules. The simplest 1040EZ tax form requires about 1,000 rules. Red Forman, associate IRS commissioner for business systems modernization, notes that "we are certain we will have tens of thousands of business rules once CADE rolls out, and that's just for individual filers" (Mosquera May 17, 2004).

In May 2003, Mark Everson was appointed IRS commissioner, and three weeks later, he appointed W. Todd Grams as the CIO. As of 2004, the project is nowhere near completion. The IRS and CSC have been repeatedly blasted by congressional reports. Relationships between CSC and the IRS are tense. CSC has been banned from participating in additional IRS projects (Perez 2004). On the other hand, in mid-2004, the FAA did hire CSC for up to \$589 million to help build an enhanced Traffic Flow Management system (McDougall 2004).

In 2006, the IRS issued more than \$318 million in refunds on phony returns because of a software failure. The IRS planned to replace the old screening system with a Web-based application by January 2006, but the organization spent \$20.5 million with no progress being developed by CSC. The IRS tried to restore the old application, but could not get it running in time. In 2005, the software caught 133,000 fake returns, stopping \$412 million in refund checks from being sent.

Without the software, the IRS halted a mere \$94 million in fraudulent refunds (Keizer 2006).

In 2007, the CADE system was operational, but not perfected. The IRS intends to use system in conjunction with its 45-year old Master File system until 2012. In 2007, the GAO accused CADE's slow processing times or delaying refunds for millions of taxpayers by several days. The CADE system also significantly exceed costs again in 2006 (Mosquera 2007).

The IRS is trying to balance a fine line when auditing people, particularly small businesses. In 2011, the IRS started demanding that small businesses turn over not just the data but also the accounting software and complete electronic files. Most small businesses use simple off-the-shelf accounting systems that do not segregate data. Providing the information requested by the IRS means turning over all of their records, including customer data. Professionals such as physicians are concerned that the request would violate HIPAA rules about patient confidentiality. But, even non-healthcare businesspeople are concerned. Benson Goldstein, senior technical manager of taxation at the AICPA noted that "Believe me, small businesses don't want the IRS calling their customers" (Saunders May 2011).

In a letter to accountants, the IRS claimed that the request was part of its "modernization" program. Chris Wagner, commissioner of the Small Business/Self-Employed Division, wrote that the audits require "unaltered metadata" so examiners can "properly consider the integrity and veracity of the electronic files." Put another way, the IRS believes that small businesses are responsible for 20 percent of the estimated \$345 billion tax gap, and it wants to completely examine business records to find any "errors" and "inconsistencies."

The Future

In the meantime, the IRS still has to process taxes. So far, electronic filing is probably the only thing keeping the agency alive. Yet if anything goes wrong with the ancient Master File system, the IRS is dead in the water. The IRS, CSC, and IBM have no choice but to get CADE running correctly as soon as possible.

Talk to citizens about paying taxes and you get lots of interesting responses. Yet a critical feature of the system is that everyone has to believe that they are being treated fairly—meaning the same as everyone else. If people somehow perceived that millions of others are getting by without paying taxes, everyone will revolt. For years, since the 1980s, the IRS has relied on a relatively simple system to automatically scan returns and identify possible tax cheats. The problem is that the rules are based on data and an economy from 20 years ago. The system is no longer catching the real tax cheats. In 2002, the IRS began collecting new data and designing new rules to identify which returns should be scrutinized more carefully. With a 13 percent increase in tax returns and a 29 percent decline in the auditing staff since 1995, the IRS has to rely on automated systems to analyze the returns. Charles O. Rossotti, the IRS commissioner at the time, could not give details of the new rules but did note that "the fact is, people who make more than \$100,000 pay more than 60 percent of the taxes, and we need to focus there" (Johnston 2002).

The electronic filing system is critical to improving productivity at the IRS. Without it, thousands of people have to enter data from the paper forms into the computer system. Yet, the existing system has several problems. Notably, it often rejects 1040 forms because of errors. The errors are anticipated, because people often make simple mistakes while entering data into their systems. The problem

is that the IRS system tends to reply with cryptic messages that users have trouble decoding. The IRS is aware of the problems, but cannot decide how to fix them. Terry Lutes, associate chief information officer for information technology services, notes that they cannot decide whether to fix the existing system or build a new one. He asks, "It's a question of how much money do you spend on a system that's going to be a throwaway?" But the drawback to a new system is that optimistically, it would not be in place until 2010 (Olsen 2004).

To top off all of the operational problems, the IRS is being criticized by the GAO because of problems with its internal accounting procedures. The GAO has been nagging the IRS about problems with their financial management system for a decade. Of 100 recommendations from the 2003 audit, the IRS has implemented only 24. The IRS is planning to address many of the other problems with another part of its modernization system, the Integrated Financial System (Mosquera April 30, 2004).

The IRS reported (2008 U.S. Budget) that electronic filing has increased from 31 percent in 2001 to 54 percent in 2006. This trend simplifies data collection and processing at the IRS, but it probably has limits.

The IRS held preliminary negotiations with vendors to outsource management and maintenance of its 100,000 desktops, but canceled the plan in late 2006. The IRS also scaled back its separate plan to outsource the handling of paper returns to a contractor. IAP Worldwide Services ran two processing centers, the other five were rescheduled for transfer after the 2006 tax year (McDougall November 2006). However, formal dates for transfers were not scheduled.

In 2007, the IRS awarded a five-year, \$9.6 million contract to General Dynamics to help manage the business systems modernization project. Although details were not provided, the role appears to be implementation-oriented instead of development (Hardy 2007).

The fiscal year 2012 budget calls for "modernizing the IRS to improve customer service and boost tax collections" (2012 Federal Budget). Most of the added money in the budget was allocated for "revenue-generating tax enforcement initiatives." Much of the budget discussion in Washington has revolved around the "tax gap," the amount of money currently being paid and the amount that Congress and the President believe people should be paying. The difference is the amount that some people and firms might be "underpaying" on their taxes. Conservatively, the president believes that better enforcement can generate \$1.3 billion a year. Part of the increased enforcement has been geared towards U.S. citizens with off-shore accounts. The IRS has been pushing people to report all of their income from other countries. In fiscal year 2010, the IRS reported that it collected \$57.60 billion through the enforcement division (www.irs.gov).

The IRS defined an IT Modernization plan in 2006 and 2007 (IRS 2007) that presents a five-year plan for improving service and enforcement. A key highlight of the problems is that many systems are based on the Master Files which were designed in the 1960s for slow computers. Most of the processing is still handled through weekly updates to the master files with programs and patches written over the course of three decades. The lack of flexibility makes it difficult to adapt to the ever-changing tax codes. On the negative side, the 2007 document notes that the BSM program proposed in 1999 could not be completed in the proposed 10-15 years because of "resource limitations." The new "plan" calls for incremental releases and leveraging existing systems. Most of the "plan" simply defined business domains and IT domains—essentially a restructuring of the IRS operations.

But, it crucially lacks any details of how the systems will be revised to support those operations. The plan does list dozens of “potential future projects” within each business domain. That is, projects that might be useful for compliance, but have not yet been designed. Interestingly, within the technical domains, converting to IPv6 seems to be an important priority. Most of the document is a laundry list of features and tasks that commercial organizations already perform.

Interestingly, the IRS Web site has links to its YouTube, Twitter, and Facebook pages. Most of the information is geared towards professional tax professionals, but the videos do contain basic information on tax related topics (IRS Web site/ New Media).

Questions

1. What problems have been experienced by the IRS in developing its information systems?
2. How are these problems related to the service’s systems development methodologies?
3. How is the IRS going to get more people to file electronically? Is there an upper limit?
4. Are there any ways to speed up the development of systems for the IRS? What would be the costs and risks?
5. Are the IRS’s problems the result of technology or management difficulties?
6. Why was the IRS unable to manage and control the CSC contract? What can the managers do differently to get the projects finished?

Additional Reading

Birnbaum, Jeffrey H., “Unbelievable! The Mess At The IRS Is Worse Than You Think,” *Fortune*, April 13, 1998.

Cranmer, Jana, “IRS Troubleshoots E-Help Program” *Government Computer News*, February 13, 2007.

Dorobek, Christopher J., “IRS: E-File Goal Is A Stretch,” *Federal Computer Week*, April 9, 2001.

Dubie, Denise, “IRS Touts Savings Project,” *Network World*, April 15, 2003.

GAO, “Tax Systems Modernization—Management and Technical Weaknesses must be Corrected if Modernization is to Succeed,” GAO/AIMD-95-156, July 1995.

Gaudin, Sharon, “IRS Fails Security Audit, 490 Computers Missing in 3 Years,” *Information Week*, April 5, 2007.

Government Computer News, “All 208 Million Tax Returns Get Electronic Scrutiny On Arrival,” March 6, 1995.

Government Computer News, “Automation Failed To Meet Requirements,” July 17, 1995, p. 6.

Hardy, Michael, “IRS taps General Dynamics for Modernization Work,” *Government Computer News*, March 12, 2007.

- Hasson, Judi, "IRS Takes On Old-Style Paperwork," *Federal Computer Week*, December 10, 2001.
- IRS, *Internal Revenue Service IT Modernization Vision & Strategy*, October 2007.
- Johnston, David Cay, "Hunting Tax Cheats, I.R.S. Vows to Focus More Effort on the Rich," *The New York Times*, September 13, 2002.
- Keizer, Gregg, "IRS Gives Away \$318 Million Because Of Bungled Software Upgrade," *Information Week*, September 5, 2006.
- Masud, Sam, "New Bells Are Ringing At IRS," *Government Computer News*, October 16, 1995, pp. 44-45.
- Mayer, Merry, "Interim Systems Will Tide IRS Over As It Modernizes," *Government Computer News*, January 25, 1999, p. 21.
- McDougall, Paul, "FAA Taps CSC to Ease Air Traffic Congestion," *Information Week*, July 2, 2004.
- McDougall, Paul, "IRS Deep-Sixes Outsourcing Plans as Tax Season Approaches," *Information Week*, November 30, 2006.
- McNamee, Mike, "A Kinder Gentler IRS?" *Business Week*, February 1, 1999, p. 128.
- "Modernization," *Government Computer News*, January 25, 1999, p. 20.
- Mosquera, Mary, "IRS Financial Management Weaknesses Linger," *Government Computer News*, April 30, 2004.
- Mosquera, Mary, "E-Filers Just Shy Of Half Of All Tax Filers," *Government Computer News*, May 10, 2004.
- Mosquera, Mary, "IRS Will Put Business Rules Engine To The Test," *Government Computer News*, May 17, 2004.
- Mosquera, Mary, "IRS Database Delay Affects Refunds," *FCW*, April 5, 2007.
- Olsen, Florence, "IRS Weighs E-Filing Upgrades," *FCW*, May 11, 2004.
- Perez, Juan Carlos, "IRS Commissioner Bars CSC from Upcoming Projects," *Computerworld*, February 13, 2004.
- Quindlen, Terrey Hatcher. "Recent IRS Award Puts TSM in Motion," *Government Computer News*, December 23, 1991, p. 3-4.
- Rosencrance, Linda, "IRS Sends Out 523,000 Incorrect Refund Check Notices," *Computerworld*, July 17, 2001.
- Rosencrance, Linda, "Accenture To Redesign, Host IRS Web Site," *Computerworld*, August 13, 2001.
- Rothfeder, Jeffrey, "Invasion of Privacy," *PC World*, November 1995, pp. 152-161.
- Saunders, Laura, "E-File or Else: What's New for Tax Season," *The Wall Street Journal*, March 19, 2011.

Saunders, Laura, "Small Businesses Fight IRS Over Data," *The Wall Street Journal*, May 26, 2011.

Smith, James M., "IRS Spends \$1b for Next Five Years Of Systems Support," *Government Computer News*, July 17, 1995, p. 82.

Thibodeau, Patrick, "Private Sector To Tackle IRS Mess," *Computerworld*, January 4, 1999, p. 4.

Varon, Elana, "For the IRS There's No EZ Fix," *CIO*, April 1, 2004.

Weiss, Todd R., "Peoplesoft Sells CRM Software to A Friendlier IRS," *Computerworld*, August 23, 2001.

Case: National Aeronautics and Space Administration (NASA)

NASA has experienced some serious problems over the past decade and a half. Some have been due to software development errors, such as the 1999 Mars Polar Lander that failed because a programmer made a calculation in miles instead of meters. Systems as complex as the ones that NASA builds are bound to have problems and most people are willing to tolerate the problems. But it turns out that rocket science must be easy compared to accounting. NASA has an ongoing problem with budgets and monitoring expenditures.

For years, NASA has routinely run over budget on projects. For years, the GAO has criticized NASA for poor accounting practices. One of the problems has been the decentralized nature of NASA—with offices scattered across the country and around the world. The financial system relied on 10 separate systems—some still written in decades-old COBOL. NASA knew it had to fix the financial reporting system, so it contracted with SAP to install their R/3 system to create a new Integrated Financial Management Program (IFMP). However, NASA has a target date of 2007 for completing the conversion, and even that number seems optimistic (GAO November 2003).

In 2004, the real nature of the mess was highly publicized by *CFO Magazine*. The article was largely driven by a disclaimed opinion by PriceWaterhouseCoopers (PwC), NASA's financial auditor. In that statement, PwC revealed that NASA's bookkeeping was so bad, that they were unable to account for \$565 billion in year-end adjustments. That is billions, not millions, of dollars. Patrick Ciganer, program executive officer for integrated financial management at NASA, explains that "we had people in each of the [10] NASA centers who knew they had to make the year-end adjustments. The problem was, they had never done them before. They had been trained, but in some cases, that was six or eight months before, and they did it wrong" (Frieswick May 2004). The data was so bad that PwC had no choice but to discredit the data and note that it could not be fixed. PwC later declined to bid for the right to audit NASA in the following year.

In echoes reminiscent of Parmalat's financial scandal, PwC also discovered that NASA was \$2 billion short in its cash account with the Treasury Department. NASA's books claimed \$2 billion that did not exist. After seven months, NASA CFO Gwendolyn Brown was still unable to find the missing money. In testimony before the House Subcommittee on Government Efficiency and Financial Management, she denied that the loss was the result of "fraud, waste, or abuse," but she had no clue about where the money went. Subcommittee chair Rep. Todd Platts (R-Pa.) retorted, "If my checkbook is off by 10 cents, I'll stay up all night until I find

that 10 cents. Your checkbook is off by \$2 billion” (Frieswick July 2004). Brown noted that the agency was going back and reconciling balances from 2000 onward. But it is unlikely that they will find the \$2 billion. PwC found several other problems involving hundreds of millions of dollars and failure by NASA to follow its own procedures and those required by the government rules.

Decentralization is a huge source of NASA’s problems. Each of the 10 centers (including Kennedy and Marshall space centers and the Glenn research center) is run independently, with separate financial systems and personnel. When Brown arrived, she attempted to integrate the financial procedures, noting, “I’ve told them that from now on, the agency will set policy, process, and procedure, and you, the centers, will do implementation. If we’re going to be accountable and credible, that’s what we have to do here” (Frieswick May 2004). One of the goals of the SAP implementation is to embed all of the central rules and consolidate the data so that all of the centers have to follow the same procedures. In particular, each center will have to reconcile its spending to the Treasury balances every month.

NASA has been extremely optimistic about implementation of the new IFMP system, or deliberately misleading—it can be hard to tell the difference. The GAO commented that “when NASA announced, in June 2003, that [the core financial] module was fully operational at each of its 10 centers, about two-thirds of the financial events or transaction types needed to carry out day-to-day operations and produce external financial reports had not been implemented in the module” (GAO November 2003). Many of the features that NASA claimed to implement had never been tested. The GAO was also concerned about NASA’s lack of oversight of its contractors and its equipment. As of 2003, NASA planned to continue using manual journal entries to handle all transactions with respect to its \$37 billion in property (GAO November 2003). About \$11 billion of that equipment is located at contractor facilities, and NASA has almost no documentation or reports on the property.

Along the same lines, NASA is unable to apply costs to specific projects. In particular, the agency is supposed to report all costs of the International Space Station (ISS). In their 2003 report, NASA did not even list the ISS—allegedly an “editorial oversight.” The GAO report indicates that the NASA system is incapable of correctly assigning costs to projects.

NASA has made several attempts to improve its accounting. The GAO notes that “NASA has made two efforts in the recent past to improve its financial management processes and systems but both of these efforts were eventually abandoned after spending a total of 12 years and a reported \$180 million” (GAO November 2003). NASA is already over budget and behind schedule with the new IFMP system. The ERP system was originally budgeted at \$982.7 million and is already \$121.8 million over budget (Dizard 2004).

By 2006, NASA was still trying to implement the SAP software. The \$1.1 billion Integrated Enterprise Management Program included a \$116 million financials upgrade. But NASA’s Office of the Inspector General found several flaws in the management and implementation of the program. It also found end-user resistance and data integration problems. End users have pointed out that the system is unnecessarily complex, does not integrate well with some existing software, and does not appear to be increasing productivity (Songini 2006).

The NASA discretionary expenditures for 2010 were \$18.912 billion or about \$187 million over budget. The agency anticipated exceeding the 2011 budget by about \$500 million. But the fiscal year 2012 proposed budget was reduced to

match the budget for 2010 or about \$18.724 billion. Most of the budget was allocated to replacing or modernizing existing buildings. Most of the space programs had already been cut, and the president stressed cooperating with private agencies and foreign nations (Russia) to provide lift capacity to space.

NASA is working to build a next generation space-based telescope. Although complex projects are difficult to forecast and to manage, an independent review panel noted that mistakes by NASA management are likely to increase the costs by \$1.5 billion. The final costs are likely to be near \$6.5 billion for a project that can be launched in 2015 at the earliest (Harwood 2010). The main problem is that the project cost estimates were too low because the initial budget did not fully understand the project, and the budgets never contained enough money to cover the costs. Chris Scolese, an associate NASA administrator commented that “Our main goal right now is to strengthen the management, which we’re doing, to strengthen the oversight, which we’re doing, and develop a good, strong estimate that we can defend. We aren’t in the business of cost overruns. We’re taking this very, very seriously.”

NASA IT has had problems with security. Of course, every hacker on the planet dreams of breaking into NASA computers. But, financial systems are equally important, both to monitor for external attacks and to ensure security for internal operations. In 2010, the Inspector General found that only 24 percent (7 of 29) systems had met Financial Information Security Management Act (FISMA) standards for annual testing. Only 52 percent (15 of 29) met the requirements for annual contingency plan testing. And only 2 of 5 of the external systems had been certified and accredited. The problems arose largely because NASA did not have an independent verification function for IT security—a common requirement in commercial sites. The CIO had purchased an information system in 2005 for \$3 million to help with the security plan, but the implementation failed and the Agency is spending money for a replacement system.

Purchases

NASA purchases billions of dollars of supplies a year. Desktop computers and accessories are important items for most of the offices. In 1998, to simplify purchases, NASA signed a nine-year \$1.3 billion deal with seven companies to provide server, desktop, communication equipment, and support services. The Outsourcing Desktop Initiative for NASA (ODIN) required the vendors to maintain inventory at each major site. The vendors created ODIN catalogs that listed all of the parts available. Some NASA sites required workers to order only from the catalogs; others allowed workers to find different deals. Normally, you would expect a centralized system to provide negotiated discounts and better prices. In this case, NASA agencies were able to save thousands of dollars by not using the catalogs. The Goddard Space Flight Center purchased 5,000 copies of antivirus software without using the catalogs and saved \$200,000. In 2003, the agency decided to tell all divisions that they were not required to purchase from the catalogs (Cowley 2003). In 2006, NASA’s basic budget was \$14.5 billion (2008 U.S. Budget). In 2007, NASA awarded a new computer-purchase contract to HP. The contract covers desktops, Blade PCs, printers, and similar products and allows up to \$5.6 billion in purchases. The contract stipulates that prices must be below the schedule prices in the U.S. General Services Administration (Singer 2007).

The Future

In terms of operations and projects, NASA is moving forward with new design ideas. In particular, NASA is working with Carnegie Mellon University on a major software dependability project. The goal is to create software that can tolerate hardware faults and security problems. These more intelligent systems would help prevent problems like those that affected the Mars Polar Lander (Thibodeau 2003).

NASA is also investing in tools to improve collaboration among researchers. Communication and information sharing have presented operational problems in the past. If everyone has the same data, it is easier to spot problems, as well as conduct new research. For the Mars rover mission, the Jet Propulsion Lab (JPL) installed the DocuShare content-management system from Xerox. Thousands of experts around the world have access to over 100,000 files. The system can handle up to 50 simultaneous projects and has search systems to help researchers find the data they need (Chourey June 2004).

NASA is also installing a new high-speed network to support operations. The system from Force10 Networks will support 10 gbps, making it possible to transfer huge files quickly. The data is carried on optical fibers. The high capacity is needed to handle even the local satellite data. Every day, NASA's 14 satellites in earth orbit transmit three terabytes of data (Chourey April 2004).

On June 20, 2004, NASA, along with the rest of the world, saw a new piece of the future: SpaceShipOne, designed and built by Burt Rutan, carried a man into space and back (Weil 2004). Partly to prove a point, partly to win the \$10 million Ansari X Prize in October, the ship did one thing extremely well. It pointed out that it is possible to reach space on a budget of a paltry \$30 million. It is not likely that Rutan's company, Scaled Composites, is going to compete with NASA anytime soon. However, it does indicate that something must be seriously wrong at NASA—when a civilian company can get to space on a budget that would not even be a round-off value in the money that NASA mysteriously loses each year.

The Space Shuttle program ended in the summer of 2011. NASA is part way through designing a new launch vehicle, but funding and direction are uncertain. Until a new system is developed, NASA will have to rely on Russian facilities to deliver people to the international space station or for other purposes. Older launch vehicles can still be used for satellite launches. Several private companies have begun developing launch capabilities, but most are currently limited to low-earth orbit. Given the federal budget crunch, and no agreement on long-term goals for NASA, the future looks somewhat bleak. On the other hand, maybe it will give NASA time to fix the computer systems.

Questions

1. How can an organization lose \$2 billion?
2. Were the auditor problems (\$565 billion) due to technology, management, individuals, or some other reason?
3. Will the new ERP system be completed? If it is, will it solve NASA's problems, particularly in terms of centralization?
4. How is NASA going to solve its problems with the development of mission software? Why is this such a challenging problem?
5. How can Congress get NASA to provide accurate progress information, and how can Congress determine if it is being told the truth?

Additional Reading

- Chourey, Sarita, "NASA to Move Data Faster With Bigger Pipes," *FCW*, April 12, 2004.
- Chourey, Sarita, "NASA Uses Xerox For Sharing," *FCW*, June 23, 2004.
- Cowley, Stacy, "NASA Wastes Money With Desktop Outsourcing Deal, Audit Finds," *Computerworld*, August 1, 2003.
- Dizard III, Wilson P., "NASA's ERP Project Goes Awry, GAO Tells Panel," *Government Computer News*, May 24, 2004.
- Frieswick, Kris, "NASA, We Have A Problem," *CFO*, May 1, 2004.
- Frieswick, Kris, "Canary Chorus," *CFO*, July 1, 2004.
- GAO, Business Modernization: NASA's Integrated Financial Management Program Does Not Fully Address Agency's External Reporting Issues, November 2003, GAO-04-151, <http://www.gao.gov/new.items/d04151.pdf>.
- Harwood, William, "NASA Management Blamed for Space Telescope Cost Overrun," *CBS News TechTalk*, November 11, 2010.
- Office of the Inspector General, *Review of NASA's Management and Oversight of Its Information Technology Security Program*, September 2010, GAO.
- Singer, Michael, "HP Lands \$5.6 Billion NASA Contract," *Information Week*, May 23, 2007.
- Songini, Marc, "NASA's SAP Launch Drags," *Computerworld*, March 27, 2006.
- Space Flight News*, "NASA's Finances In Disarray; Auditor Quits," May 16, 2004.
- Thibodeau, Patrick, "NASA Reinvents Troubled IT With Help of Private Sector," *Computerworld*, February 10, 2003.
- Weil, Elizabeth, "Rocketing Into History," *Time*, June 23, 2004.

Summary Industry Questions

1. What information technologies have helped this industry?
2. Did the technologies provide a competitive advantage or were they quickly adopted by rivals?
3. Which technologies could this industry use that were developed in other sectors?
4. Is the level of competition increasing or decreasing in this industry? Is it dominated by a few firms, or are they fairly balanced?
5. What problems have been created from the use of information technology and how did the firms solve the problems?