Systems Analysis and Design

WEEK
2

Many programmers try to sit down and write a program without any preplanning. Programmers who write successful programs tend to follow a more structured approach. Today you will learn:

- ☐ Several approaches to creating a complete program.
- ☐ Why you should follow a structured approach when creating a program.
- ☐ What is meant by structured systems analysis and design.
- ☐ About prototyping.
- ☐ How to apply some of the systems analysis and design concepts to an actual application.

Creating a Computer System

Many programmers believe that the way you create a computer program is at the keyboard. You get the idea, you think about it for a few minutes (or longer), and then you start keying in the code. For many programmers, this works for awhile; however, it poses several problems.

The most common problem with this method of programming is maintainability. Typically, programs that have been written without first thinking through what is being developed are disorganized. They may appear structured and clean to the user, but the underlying code is generally tangled and patched.

Most—if not all—successful corporations or programming development shops won't allow programs to be written without forethought. Successful companies generally follow a methodology when developing programs. A methodology is a highly glorified term that has a simple meaning. A *methodology* is a set of procedures that are followed to accomplish a task. Methodologies for creating computer programs generally follow a set of systems analysis and design procedures. These procedures cover what should be done from the time an idea for a program or system is thought of until the time it is complete and turned over for use.

Note: A computer system differs from a computer program. A *computer system* may include programs, procedures, hardware, and more. A *computer program* is simply software.

There are several methodologies that are used within the development of computer programs. Three will be briefly covered today. The first is structured systems analysis and design. This is a methodology that is equipped to handle the biggest or even the smallest of computer programs or systems. The second methodology covered is a form of rapid or accelerated systems development. This is a scaled-down version of structured systems analysis and design that is intended to enable you to develop a system quicker, yet still follow the structured approach. The third methodology covered is prototyping. This is the methodology that a large number of PC programmers choose to follow.

Understanding these three methodologies will help you in your approach of the development of a computer system of any size. After these three methodologies are covered, you will be introduced to a fourth approach. This fourth method uses parts of the other three. It is aimed more directly at smaller PC-developed systems. This fourth methodology is what has been applied to the application that will be developed throughout the rest of this book. As you develop your own systems, you'll need to decide which methodology best suits your needs.

The Traditional Approach: Structured Systems Analysis and Design

The traditional approach is the most commonly used in businesses. It is broken down into five stages. Each of these stages has specific procedures that need to be completed. In addition, each stage follows the preceding stage. If a step is missed in a preceding stage, then you should go back and redo everything from that point. The five stages are:

Stage 1: Preliminary Analysis
Stage 2: Systems Analysis (Logical Design)
Stage 3: Systems Design (Physical Design)
Stage 4: Construction
Stage 5: Wrap Up (Testing and Implementation)

Each of these stages is also broken down into specific steps that should be followed. It's the structure of the stages and steps, along with the specific order, that gives this approach its name.

In addition to the specific steps, structured systems analysis and design has several documents that get produced. The documentation that is produced through this methodology won't be covered in this book. There are complete books and several college courses available on this methodology.

Stage 1: Preliminary Analysis

The first stage of structured systems analysis and design is the preliminary analysis. In this stage, the problem is defined and the overall direction is set. This stage generally starts with someone presenting an idea. To help formulate the idea, a preliminary study is done. This study is used to determine if the system (or program) is feasible. The following steps are performed in the preliminary study:

Step 1: Overview the problem/objective
Step 2: Set the objectives/capabilities
Step 3: Determine the scope of the project
Step 4: Determine the business rules/assumptions/constraints
Step 5: Do preliminary estimates
Step 6: Scheduling

Step 1: Overview the Problem/Objective

A statement of what is intended to be accomplished or what problem needs to be solved should be made. This should be composed in a few sentences to a couple of paragraphs. This should be a very high-level description of what the system will do.

Step 2: Set the Objectives/Capabilities

The overall goals and capabilities of the system should be listed. These should be presented in a bulleted list that describe not what the system is, but what the system should be. By listing the capabilities and objectives, you begin to define what the system will need to do. If you are developing a system similar to something that already exists, then you should state what your system will do differently. Examples of objectives include:

- ☐ System should help save time.
- ☐ System should provide information so that decisions can be made faster.
- ☐ System should help provide more accurate information.
- ☐ System should be easy to use so that learning time is short.

Note: The objectives should state what your system or program will offer that other programs don't already offer.

Step 3: Determine the Scope of the Project

The scope of the project refers to the size of the project. It's the scope of a project that prevents it from getting too large. If a project doesn't have a scope, then there is nothing to prevent it from growing forever. For example, consider a program that lists a text file. The only objective of the program is to be able to list the program to the screen or to the printer. Without a scope, there are several solutions to solve just such a problem. The simplest solution would involve simply printing a listing as it appears to either the printer or the screen; however, there are a variety of solutions that would be much more interesting. You could create a system that displays the file on the screen and enables you to view it before selecting a menuing option to print it. The printing could enable you to select only a portion of the text or all of it. In addition, the system could enable you to edit the program before printing. The possibilities can grow until they get out of hand. In addition, what could have been a simple command at the operating system level may become a system that requires several programmers many months to create.

With scope, you determine the limits of the program. You can set what information is needed, what equipment may—or may not—be used, what the budget is, and the amount of time that is available. Anything that can cause limitations on the final system should be included.

Step 4: Determine the Business Rules/Assumptions/Constraints

In addition to scope factors, there are also business rules and other rules that can constrain a system. Such rules may include company policies and practices, equipment limitations, or cost factors. For example, you can't effectively develop a UNIX system if you don't have UNIX. In addition, you can't develop a virtual reality operating system if you don't have the hardware to work with virtual reality.

Step 5: Do Preliminary Estimates

Once you have an idea of the objectives and limitations, you can make a first attempt at determining the costs and benefits of the system or program. Costs come in several forms such as the software and licensing that may be required, and the time to develop and program.

These costs should be compared to the benefits that will be achieved to determine if it's worth proceeding with the system. Many of the possible benefits can be seen in the objectives listed earlier. An example of cost versus benefits can be seen in the development of a calculator program for Microsoft Windows. Because Windows comes with a calculator built in, people aren't going to spend money to buy a new one unless it offers something extra. If the cost of developing the new calculator is more than what you think you can make selling it, then it may be a bad project—unless there are other benefits that the user will receive.

Step 6: Scheduling

Like the functionality of a system or program, the amount of time can also get out of hand. To let others know when they can expect the new program or system, a schedule needs to be set. This schedule should include time frames for the rest of the Structured Systems Analysis and Design steps.

Generally, Project Management Tools are used to schedule large systems. Such tools include Project Workbench, Super Project, Project Scheduler, and more. These programs enable you to make estimates on when tasks should be accomplished. They then chart your progress to show when you are on time and when you are behind. They can also chart the relationships between tasks—which tasks must be accomplished before others. Such tools help make managing your time and the project much easier.

Stage 2: Systems Analysis (Logical Design)

The steps in the preliminary study should be done at a very high level and in a relatively quick time frame. Their purpose is simply to set the stage for the detailed research that will be performed on the system. In addition, they should provide enough information so that a decision on continuing the system can be made.

In the second stage, systems analysis, the level of detail becomes much more detailed. This involves the following four steps:

Step 1: Refine the problem/create team
Step 2: Determine the facts
Step 3: Provide alternatives
Step 4: Determine the next step

Step 1: Refine the Problem/Create Team

Refining the problem means you need to ensure that what was stated in the Preliminary Design was accurate. The information gathered should be reviewed to ensure that it sets the proper direction.

If the direction is accurate, then a team must be pulled together to analyze and construct the system. The team may be you alone or with more people. If there are several people working on the project, then the roles that these people are going to serve should be defined.

Step 2: Determine the Facts

Once you determine who is on the project, it's time to begin the detailed analysis by gathering the facts. The facts are any information that is related to what needs to be accomplished by the system. In addition, this includes the details of what must be accomplished. Specifics are determined. These specifics fall into three areas: what comes into the system, what leaves the system, and what does the system do in between receiving information and providing it. In addition to these three areas, the volume of information that will be used by the system needs to be determined. You may find that what you are attempting to do is impossible.

For example, if you are creating a database to track your contacts, then you could have several inputs and outputs. The inputs may include photographs, demographic information, a history of contacts, and more. The outputs may include a number of different types of inquiries into the information. You should estimate how much room it will take to store the information for each contact. In addition, you should estimate the number of contacts that will be tracked. If each contract is expected to take 1MB of storage space (with a photo and history information this is possible), and if you are expecting 1,000 contacts, then you have a system that requires a gigabyte of data storage. Your business rules may state that this is not possible.

Consider Logical, Not Physical

In doing the systems analysis, you should consider the logical nature of the system, not the physical. Physical considerations are those that assume specific computer programs, specific computer hardware, specific storage mediums, specific display

mediums, or any other physical items that will be used. You should only consider the logical information by concentrating on what needs to be done and why. For example, if a system such as the one mentioned previously is to capture contact information, then you consider the logical aspects. What information needs to be captured, not how it needs to be captured. What information needs to be accessed, not whether it's displayed on a report or on the screen. The physical characteristics will be designed in the next stage of structured systems analysis and design.

Consider Available Tools

There are several tools that have been developed to help pull together the facts and to help document them. The tools include flow charts and CASE tools. Flow charting, data flow diagrams, structure charts, action diagrams, input/output models, context diagrams, and entity-relationship models are all different methods for documenting different parts of a system. Books have been written on most of these forms of documentation.

Pull the Facts Together

A systems analysis document should be created when all the facts have been gathered. This document should contain detailed information on the logical flow and design of the system. It should contain all of the charting that may have been done with analysis tools along with descriptions of the data that comes into and leaves the system. It also should contain the process that acts on the data.

Step 3: Provide Alternatives

Once the detailed analysis has been performed, you should consider your alternatives. Most people assume they need to develop a new system on their own; however, this is a poor assumption. There may be a number of systems already out there doing the same thing you are attempting to do. It may also be that what you are attempting to do is not feasible. In this case, your only option may be to kill the project. Alternatives should be determined. Generally, the alternatives fall into the following four areas:

1. Use an off-the-shelf package.
2. Modify an off-the-shelf package.
3. Create your own new system.
4. Cancel the system and do nothing.

Expert Tip: Determining if an off-the-shelf package is the right solution is not always easy. You may need to talk to software dealers, visit local software stores, and consult trade publications. You should look for packages that come close to meeting the needs that you have detailed.

Step 4: Determine the Next Step

Before you can continue to the next stage, you must make a decision. You must pick one of the alternatives that you stated in the previous step. In corporations, the user(s) of the system would pick the alternative. If the alternative is to modify a package or to create your own system, then you would continue on to the next stage of structured systems analysis and design.

Stage 3: Systems Design (Physical Design)

At this point, you know what needs to be done and you now have to design the physical aspects of the system. This includes the way information will be stored, accessed, and processed.

Note: A *prototype* is a mock-up of what a screen or report will look like.

This is all done with tools similar to those used in the systems analysis stage. Most of the outputs from systems analysis should be used as the starting point of the design. This stage involves creating file layouts, screen prototypes, and more. The physical nature of the system also is detailed. In addition, decisions on the hardware to be used in the system will need to be made. These include decisions such as whether reports appear on paper, the screen, or both. The specific information that will appear in the reports should be detailed; and screen prototypes should also be developed.

Stage 4: Construction

This is most people's favorite stage. Most people think of construction as the coding stage. While it is the coding, construction also includes tasks that help in coding.

These tasks include creating structure charts, writing pseudo-code, and flowcharting. These again are tasks that help organize and identify problems before the coding starts. Books have been written on methods and rules for accomplishing these tasks.

When this stage is done, the programs should be coded and ready for use; however, the project is not done. Before the project is done, the fifth stage (Wrap Up) should be completed.

Stage 5: Wrap Up (Testing and Implementation)

This is the most overlooked stage of structured systems analysis and design. This stage contains the following steps that can make or break a software program or system:

Step 1: Testing
Step 2: Documenting
Step 3: Implementing

Step 1: Testing

Testing is often neglected by most software developers. Once the program is completed and appears to work, it is passed on. Programs stand a better chance of success if they are tested. The two levels of testing that are generally performed are unit testing and integration testing.

Unit testing means testing individual programs. A word processing package may be composed of several programs—a spell checker, a thesaurus, an entry program, a printing program, and more. The users of the word processing package may not be aware that several programs are running. To them, it is a single, complete system. In unit testing, each of these pieces is tested on its own to ensure that it is complete.

Integration testing tests the system as a whole. Integration testing ensures that all the individual pieces work together. In addition, integration testing ensures that the programs work with the operating system and other programs that may be running.

By doing both integration and unit testing, you help ensure that a program is error free. In addition, by setting up these tests, you can ensure that the system works properly. If there is a problem, once it is fixed, not every test needs to be rechecked. You need to rerun only the unit test for the changed code. You should also rerun the integration tests to ensure that no interactions have been affected.

Step 2: Documenting

Any user manuals or other documentation should be created at this point. Because structured systems analysis and design was used, there should be a large amount of documentation already created. In this stage of the project, the documentation should be reformatted, and user manuals and other documents can be created.

Step 3: Implementing

When testing is done, a system is ready to be implemented. Implementation may involve preparing the software to be sold, setting up a system, uploading the software to a bulletin board as shareware, or installing it on your own computer. The documentation should also be provided.

All the steps of structured systems analysis and design should enable you to implement a system that works almost exactly as needed. There should be no surprises. By following all of the steps, you should have completed the best system based on the objectives and restrictions.

Note: This has been a very high-level discussion of structured analysis and design. The main purpose of this discussion is to make you aware of the complexity that can be involved in developing large computer systems or programs. If you were attempting to create a software package similar to one of the larger packages on the market (for example, Paradox, WordPerfect, Excel, etc.), you would want to use a methodology such as structured systems analysis and design to control the size. For smaller systems, other methodologies may be better.

DO	DON'T
DO understand the overall idea of structured systems analysis and design.	
	DON'T confuse logical design with physical design.
DO consider other books if you are interested in more information on structured systems analysis and design.	

Rapid or Accelerated Systems Analysis and Design

Rapid or accelerated systems analysis and design was developed to help create a system quicker. This methodology follows what was presented in structured systems analysis and design; however, there are a few subtle twists.

The first is that prototyping is done as early as possible. As soon as you have an idea of what is going to be created, you create prototypes. These prototypes are dynamic throughout the analysis and design. As new information is learned, new iterations, or new versions, of the prototypes are created. In addition, alternatives are eliminated. You make the assumption that you are creating a new system. As soon as there is a high comfort level with the prototypes, construction begins.

Expert Tip: There is no set number of versions that the prototypes will go through. You should continue updating the prototypes until there is a high comfort level.

This methodology has a greater chance for failure than the structured approach. Because prototypes are created early on, there is a tendency to lock into designs early on. In addition, while most of the steps in the structured method are followed, there is a tendency to leave out as many as possible. Because of this and the accelerated nature of this methodology, objectives or business rules may be overlooked.

Prototyping

Most at-home PC developers use prototyping if they use any methodology at all. Prototyping ignores most, if not all, of the analysis and design. Screen prototypes and report prototypes are developed. Once developed, construction begins. While this methodology isn't much better than no methodology at all, it does provide some chance for thinking a system through before coding begins.

Several tools are available for prototyping. One of the more popular screen prototyping tools is Demo II. Some editors will also enable you to prototype screens. For prototyping reports, text editors generally will work fine.

DO	DON'T

DO at least prototype your system before you begin it.

A Program Specification

In developing smaller PC systems, I use a methodology that is different from all of those presented so far. It is also similar to that used by several of the companies that I have worked in. This methodology involves developing a single document—a program specification—that serves as an overview for the system, or part of a system, to be developed. In developing this specification, concepts presented in the structured systems analysis and design are used. In addition, a large portion of the specification revolves around the prototypes.

A program specification should have the following parts:

- ☐ An overview/purpose
- ☐ Capabilities/objectives
- ☐ Business rules/assumptions/limitations
- ☐ Processing components
- ☐ Prototypes
- ☐ Input/output requirements

Most of the remaining days that you spend reading this book revolve around developing a single application. A specification for this application has been created. This specification follows today's material. You should use it as a template for creating your own specifications. This specification doesn't contain information on the entire system that will be developed. The reporting sections of the application will be contained in their own specification later in the book.

Although most of the specification follows what is presented in structured analysis and design, a few comments bear mentioning. You should complete the first three sections before creating a prototype. These sections help layout an overview of the program(s).

When the first three sections are completed, the prototype can be developed. The prototype can then be used to fill in the processing components. The processing components detail the actions that are available on each screen. Each key is detailed.

In addition, data matrixes may be used. These can detail descriptions of the fields on entry screens. They can include information on the data types, the field sizes, and the edits that should be performed. If there are special edits, they may need to be detailed in the processing components section of the specification, rather than the data matrix.

The input/output requirements detail the database access that the program is going to require. This helps you to later determine the appropriate structure for your program's files.

Tracking Questions

One addition that you may choose to add to your specification is a question and answer section. If you are working with others in the system that you are creating, it's often important to track questions. It's not uncommon for users to change their mind on what they want. As you ask questions, or as users change their minds, you may wish to track this. A good place to place these questions and answers is after the business rules and before the processing components.

DO	DON'T
DO review the specification following today's lessons. It details the program that will be developed throughout this book.	
DO track questions and answers if you are working with users who change their minds a lot.	

Summary

Today's material covered a very important advanced programming topic. Methodologies for developing computer systems were presented. By using a methodology, you have a better chance of developing a system that will be truly useful. Four different approaches were covered. The first, structured systems analysis and design, is the most detailed. Rapid or accelerated systems analysis and design is a similar approach that works at a faster pace by using prototypes. Prototyping, the third methodology presented, enables you to create mock-ups of your program's screens and reports before coding begins. Program specification was the fourth approach presented. A program specification has been created that details the entry portion of the application that will be developed in the rest of this book. You should review this specification.

Q&A

Q What percentage of structured systems analysis and design is spent coding programs?

A If done correctly, only a small portion of time will be spent actually coding a system. Generally, 20 to 30 percent of the time spent on the project should be in coding. This may seem strange, however, because of the pre-thinking provided by the methodology, the time spent coding will be extremely focused on the correct tasks.

Q Why is following a methodology important?

A In following a methodology—even a scaled down version of a methodology—you are pre-planning. It's easier to make corrections before you start coding than after. Once you start coding a system, a change can be harder to make. Generally, a change requires much more work if it is made after coding has started. Changes caught before programming beings are easier to implement. If a large number of changes occur after contraction begins, the code can be come convoluted.

Q What is the difference between logical design and physical design?

A Logical design should come first. Logical design evaluates what a system should do. Physical design evaluates how a system should do the what. Physical design deals with the hardware and software that may be needed, where as the logical design will only deal with information.

Q What is object-oriented design?

A Where as structured systems analysis and design starts with an overview and focuses later on the detail; object-oriented design starts with the data and builds out toward the entire systems. This is an extremely high-level definition. For more information, consider reading a book on systems analysis and design or a book specifically on object-oriented design.

Workshop

The Workshop provides quiz questions to help you solidify your understanding of the material covered and exercises to provide you with experience in using what you've learned.

Quiz

1. What is a methodology?
2. What is meant by scope?
3. What is a prototype?
4. Should a methodology be followed?
5. What is a reason for not following a methodology?

Exercises

1. Review the program specification in the following section. Look at each section to see what information is presented.
2. After looking at the following specification, create your own. Create a specification for a similar type of application that allows for the entry of information. You can use this specification to create your own application during the following days.