QUALITY ASSURANCE AND IMPLEMENTATION

PART 2: SYSTEM DESIGN AND DEVELOPMENT TECHNIQUES

Dr. Krishnendu Guha

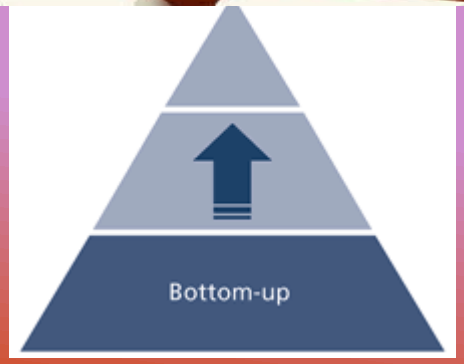
Assistant Professor (On Contract)

National Institute of Technology (NIT), Jamshedpur

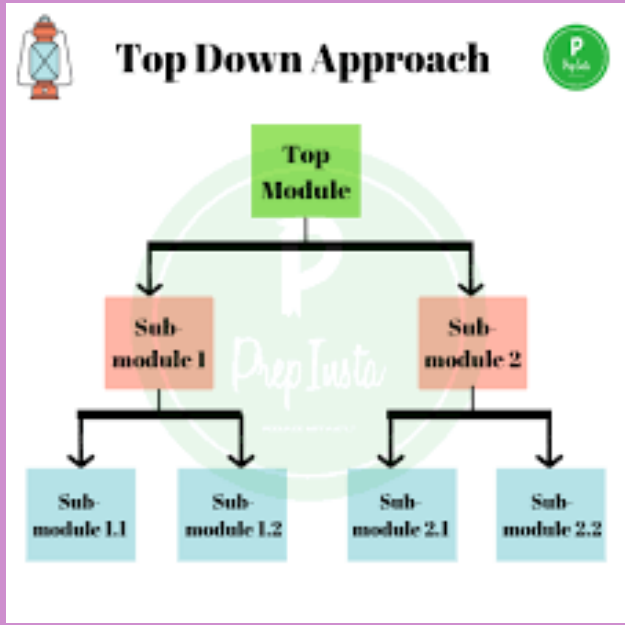
Email: krishnendu.ca@nitjsr.ac.in

TOP-DOWN SYSTEMS DESIGN AND DEVELOPMENT

- Businesses often take this approach to systems development by going out and acquiring, for example, COTS software for accounting, a different package for production scheduling, and another one for marketing.
- When in-house programming is done with a bottom-up approach, it is difficult to interface the subsystems so that they perform smoothly as a system.
- Interface bugs are enormously costly to correct, and many of them are not uncovered until programming is complete, when analysts are trying to meet a deadline in putting the system together.
- At this juncture, there is little time, budget, or user patience for the debugging of delicate interfaces that have been ignored.

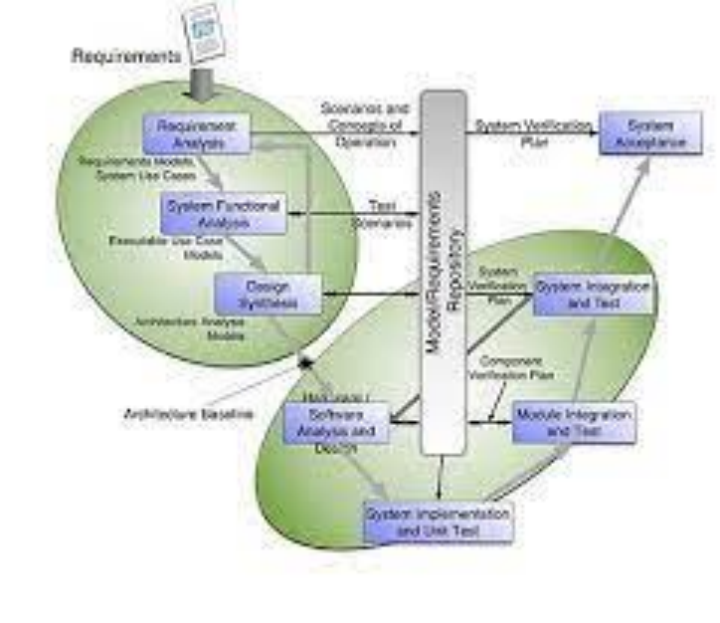


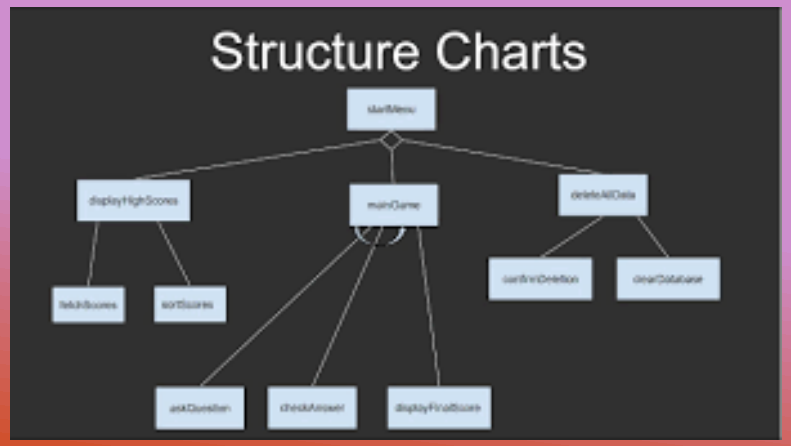
- Limitations to taking a bottom-up approach.
- One is that there is a duplication of effort in purchasing software and even in entering data.
- Another is that worthless data are entered into the system.
- A third, and perhaps the most serious drawback of the bottom-up approach, is that, while pockets of users' needs may have been met, overall organizational objectives are not considered and hence cannot be met.



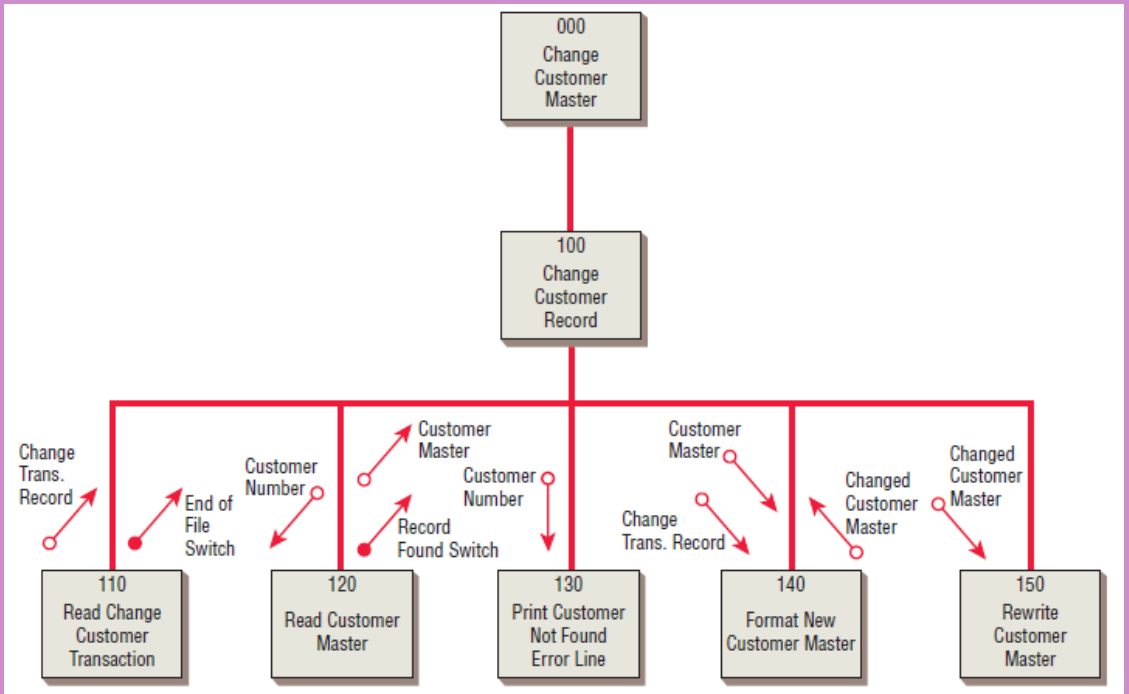
- Top-down design allows the systems analyst to ascertain overall organizational objectives first, as well as to ascertain how they are best met in an overall system.
- Then the analyst divides that system into subsystems and their requirements.
- When systems analysts employ a top-down approach, they are thinking about the interrelationships and interdependencies of subsystems as they fit into the existing organization.
- The top-down approach also provides desirable emphasis on synergy or the interfaces that systems and their subsystems require, which is lacking in the bottom-up approach.
- It helps to answer the question of how teams must work together to accomplish their goals.
- The advantages of using a top-down approach to systems design include avoiding the chaos of attempting to design a system all at once.
- As we have seen, planning and implementing management information systems is incredibly complex. Attempting to get all subsystems in place and running at once is agreeing to fail.
- A second advantage of taking a top-down approach to design is that it enables separate systems analysis teams to work in parallel on different but necessary subsystems, which can save a great deal of time.
- A third advantage is that it prevents systems analysts from getting so mired in detail that they lose sight of what the system is supposed to do.

USING STRUCTURE CHARTS TO DESIGN MODULAR SYSTEMS





- Once the top-down design approach is taken, the modular approach is useful in programming.
- This approach involves breaking the programming into logical, manageable portions, or modules.
- This kind of programming works well with top-down design because it emphasizes the interfaces between modules and does not neglect them until later in systems development.
- Ideally, each individual module should be functionally cohesive so that it is charged with accomplishing only one function.
- Modular program design has three main advantages.
- First, modules are easier to write and debug because they are virtually self-contained. Tracing an error in a module is less complicated, because a problem in one module should not cause problems in others.
- A second advantage of modular design is that modules are easier to maintain. Modifications usually will be limited to a few modules and will not be spread over an entire program.
- A third advantage of modular design is that modules are easier to grasp, because they are self-contained subsystems. Hence, a reader can pick up a code listing of a module and understand its function.

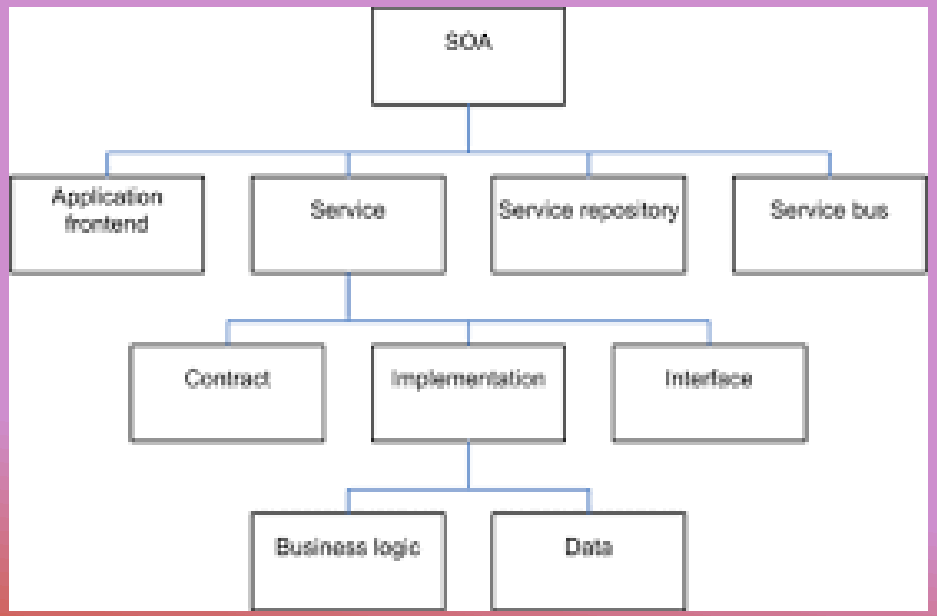


a set of modules used to change a customer record, shows seven modules that are labeled 000, 100, 110, 120, and so on. Higher-level modules are numbered by 100s, and lower level modules are numbered by 10s. This numbering allows programmers to insert modules using a number between the adjacent module numbers. For example, a module inserted between modules 110 and 120 would receive number 115.

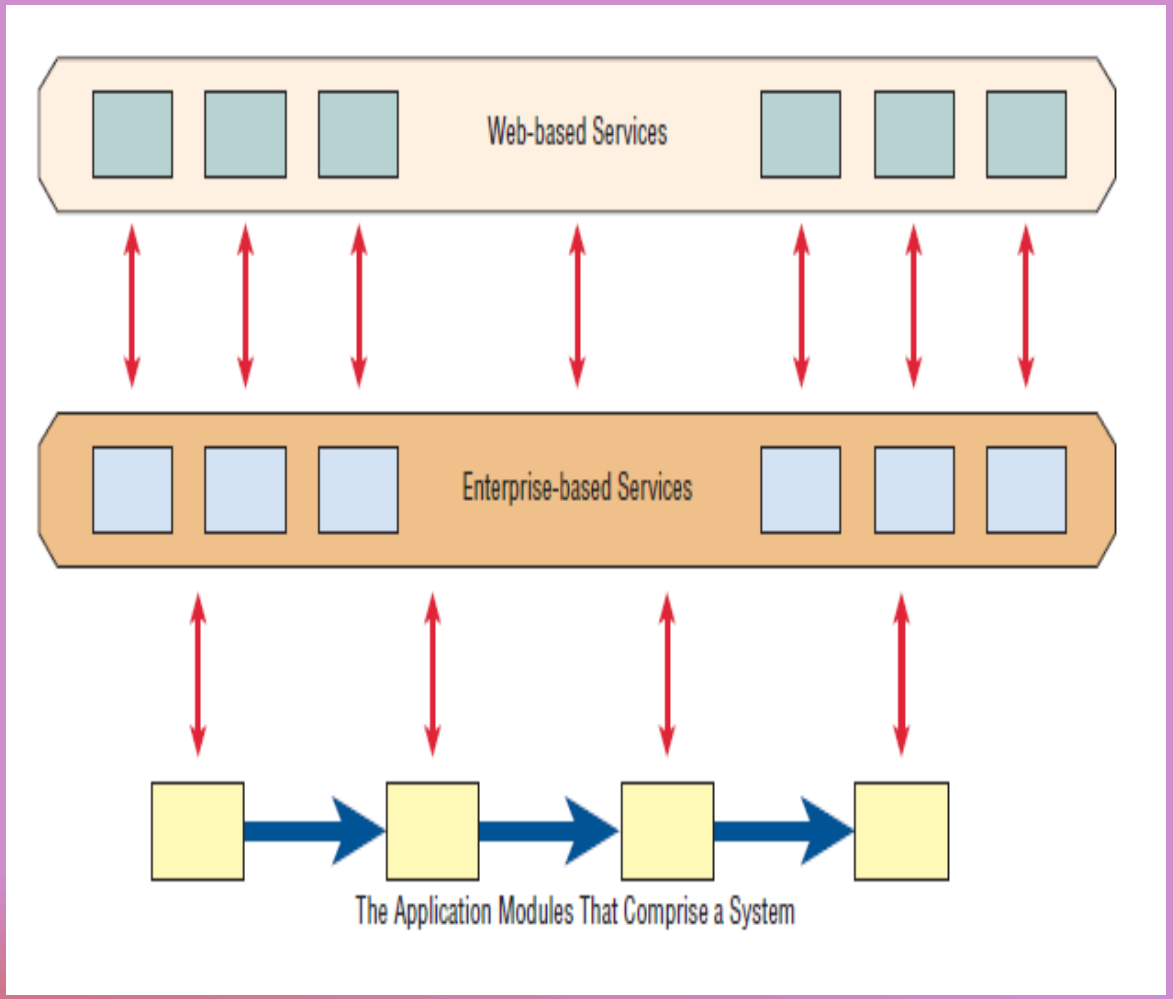
- Some guidelines for modular programming include the following:
- **1.** Keep each module to a manageable size (ideally including only one function).
- **2.** Pay particular attention to the critical interfaces (the data and control variables that are passed to other modules).
- **3.** Minimize the number of modules the user must modify when making changes.
- **4.** Maintain the hierarchical relationships set up in the top-down phases.

- The recommended tool for designing a modular, top-down system is called a structure chart.
- A structure chart is simply a diagram consisting of rectangular boxes, which represent the modules, and connecting arrows.
- Off to the sides of the connecting lines, two types of arrows are drawn. The arrows with the empty circles are called data couples, and the arrows with the filled-in circles are called control flags or switches. A switch is the same as a control flag except that it is limited to two values: either yes or no. These arrows indicate that something is passed either down to the lower module or up to the upper one.

SERVICE-ORIENTED ARCHITECTURE (SOA)



- Modular development has led to a concept called service-oriented architecture (SOA), but one that is very different from the modules in the structure chart.
- Instead of being hierarchical like the top-down approach found in structure charts, the SOA approach is to make individual SOA services that are unassociated or only loosely coupled to one another.
- Each service executes one action.
- One service may return the number of days in this month; another may tell us if this is a leap year; a third service may reserve five nights in a hotel room from the end of February to the beginning of March. Although the third service needs to know the values obtained from the first and second services, they are independent of one another.
- Each service can be used in other applications within the organization or even in other organizations.
- We can say that service-oriented architecture is simply a group of services that can be called upon to provide specific functions.
- Rather than including calls to other services, a service can use certain defined protocols so that it can communicate with other services.



The Application Modules That Comprise a System

- Services can be general in nature and can be outsourced or even be available on the Web.
- Other services are more specialized and oriented toward the business itself. These enterprise-based services provide business rules and can also differentiate one business from another. Services can be called upon at a time and can be called on repeatedly in many application modules.
- The burden of connecting services in a useful fashion, a process called orchestration, is placed upon the systems designer. This can even be accomplished by selecting services from a menu of services and monitoring them by setting up an SOA dashboard.
- In order to set up an SOA, the services must be:
- **1.** modular,
- **2.** reusable,
- **3.** work together with other modules (interoperability),
- **4.** able to be categorized and identified,
- **5.** able to be monitored, and
- **6.** comply with industry-specific standards.
- While the advantages of reusability and interoperability are obvious, SOA is not without its challenges.
- First industry standards must be agreed upon.
- Next, a library must be maintained so that developers can find the services they need.
- Finally, security and privacy can be issues when using software developed by someone else.

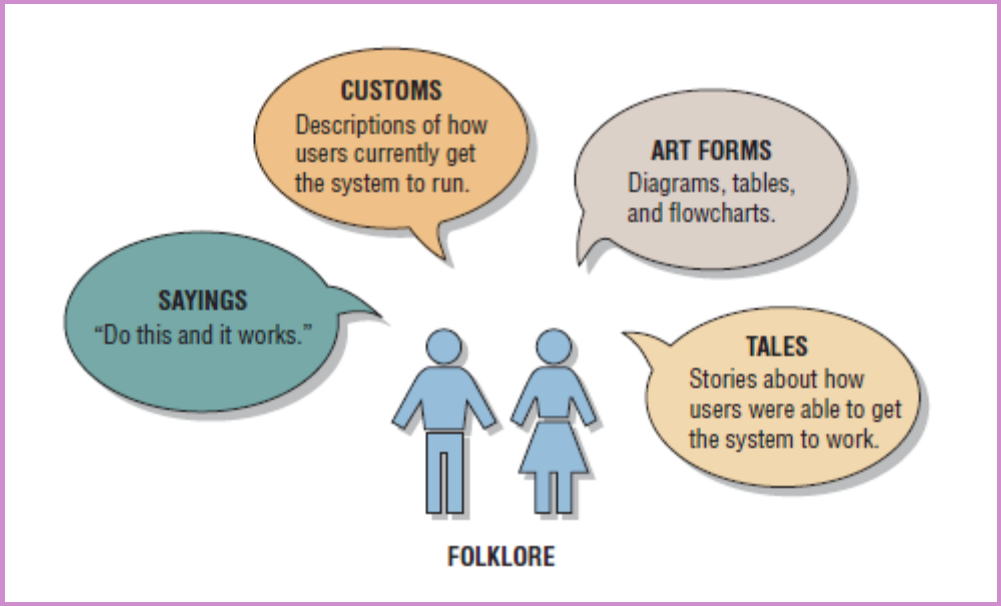
DOCUMENTATION APPROACHES

- The total quality assurance effort requires that programs be documented properly.
- Software, systems, and formal and informal procedures need to be documented so that systems can be maintained and improved.
- Documentation allows users, programmers, and analysts to “see” the system, its software, and procedures without having to interact with it.
- Turnover of information service personnel has traditionally been high in comparison with other departments, so chances are that the people who conceived of and installed the original system will not be the same ones who maintain it.
- Consistent, well-updated documentation will shorten the number of hours required for new people to learn the system before performing maintenance.
- There are many reasons why systems and programs are undocumented or under-documented.
- Systems analysts may fail to document systems properly because they do not have the time or are not rewarded for time spent documenting.
- Some analysts do not document because they dread doing so or think it is not their real work.
- Furthermore, many analysts are reticent about documenting systems that are not their own, perhaps fearing reprisals if they include incorrect material about someone else’s system.
- Defenders of the SDLC approach remind us that documentation accomplished by means of a CASE tool during the analysis phases can address many of these problems.

PROCEDURE MANUALS

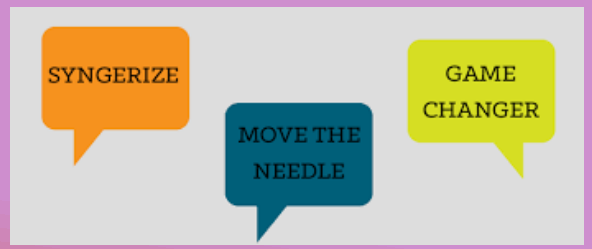
- Procedure manuals are common organizational documents that most people have seen.
- They are the English-language component of documentation, although they may also contain program codes, flowcharts, and so on.
- Manuals are intended to communicate to those who use them.
- They may contain background comments, steps required to accomplish different transactions, instructions on how to recover from problems, and what to do next if something isn't working (troubleshooting).
- Use of the Web has revolutionized the speed with which assistance can be obtained by users.
- Many software developers have moved user support—complete with manuals, FAQ pages, online chat, and user communities—to theWeb.
- Key sections of a manual should include an introduction, how to use the software, what to do if things go wrong, a technical reference section, an index, and information on how to contact the manufacturer.
- The biggest complaints with procedure manuals are that
- (1) they are poorly organized,
- (2) it is hard to find needed information in them,
- (3) the specific case in question does not appear in the manual, and
- (4) the manual is not written in plain English.

THE FOLKLORE METHOD



Customs, tales, sayings, and art forms used in the FOLKLORE method of documentation apply to information systems

- FOLKLORE gathers information that is often shared among users but is seldom written down.
- FOLKLORE was first developed in the 1980s by Kendall and Losee, well before the creation of blogs and user communities.
- FOLKLORE has two main advantages over commonly found user communities:
- (1) it is structured, resulting in more organized, more complete documentation, and
- (2) it encourages someone familiar with the software to seek out information rather than depending on users to come forth on their own.
- FOLKLORE is a systematic technique, based on traditional methods used in gathering folklore about people and legends.
- This approach to systems documentation requires the analyst to interview users, investigate existing documentation in files, and observe the processing of information.
- The objective is to gather information corresponding to one of four categories: customs, tales, sayings, and art forms.



- When documenting customs, the analyst (or other folklorist) tries to capture in writing what users are currently doing to get all programs to run without problems. An example of a custom is: “Usually, we take two days to update the monthly records because the task is quite large. We run commercial accounts on day one and save the others for the next day.”
- Tales are stories that users tell regarding how the system worked. The accuracy of the tale, of course, depends on the user’s memory and is at best an opinion about how the program worked.
- Tales normally have a beginning, a middle, and an end. So we would have a story about a problem (the beginning), a description of the effects (the middle), and the solution (the end).
- Sayings are brief statements representing generalizations or advice.
- We have many sayings in everyday life, such as “April showers bring May flowers,” or “A stitch in time saves nine.” In systems documentation, we have many sayings, such as “Omit this section of code and the program will bomb,” or “Always back up frequently.” Users like to give advice, and the analyst should try to capture this advice and include it in the FOLKLORE documentation.
- Gathering art forms is another important activity of traditional folklorists, and the systems analyst should understand its importance, too.
- Flowcharts, diagrams, and tables that users draw sometimes may be better or more useful than flowcharts drawn by the original system author. Analysts will often find such art posted on bulletin boards, or they may ask the users to clean out their files and retrieve any useful diagrams.
- Contributors to the FOLKLORE document do not have to document the entire system, only the parts they know about.
- Just like with Web-based user communities, the danger of relying on FOLKLORE is that the information gathered from users may be correct, partially correct, or incorrect.

CHOOSING A DESIGN AND DOCUMENTATION TECHNIQUE

- The techniques discussed in this chapter are extremely valuable as design tools, memory aids, productivity tools, and as a means of reducing dependencies on key staff members.
- The systems analyst, however, is faced with a difficult decision regarding which method to adopt. The following is a set of guidelines to help the analyst use the appropriate technique.
- Choose a technique that:
- **1.** Is compatible with existing documentation.
- **2.** Is understood by others in the organization.
- **3.** Allows you to return to working on the system after you have been away from it for a period of time.
- **4.** Is suitable for the size of the system on which you are working.
- **5.** Allows for a structured design approach if that is considered to be more important than other factors.
- **6.** Allows for easy modification.


Thanks
THANK YOU!